

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

Benjamin Dowling¹, Paul Rösler², and Jörg Schwenk²

¹ Applied Cryptography Group, Eidgenössische Technische Hochschule Zürich
benjamin.dowling@inf.ethz.ch

² Horst-Görtz Institute for IT Security,
Chair for Network and Data Security, Ruhr University Bochum
{paul.roesler,joerg.schwenk}@rub.de

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 Noise 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 to 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). In contrast, Noise allows the transmission of encrypted messages as soon as any key is established (for instance, before authentication between parties has taken place), and then incrementally increases the channel's security guarantees. By proposing a generalization of the original ACCE model, we capture security properties of such staged channel establishment protocols flexibly – comparably to the multi-stage key exchange model (Fischlin and Günther CCS 2014).

We give security proofs for eight of the 15 basic Noise patterns.

Keywords: Noise protocol framework; ACCE; Multi-Stage; Channel Establishment

Table of Contents

1	Introduction	3
1.1	Related Work	7
2	Preliminaries	9
2.1	Notation	9
2.2	The PRF-Oracle-Diffie-Hellman Assumption	9
3	The Noise Protocol Framework	10
3.1	Noise Protocol Patterns	12
4	Replay Attacks, State Reveals, and Their Relation	14
5	Flexible ACCE Framework	16
5.1	fACCE Primitive Description	17
5.2	Execution Environment	18
5.3	Flexible Security Notion	20
5.4	Adversarial Model	22
5.5	Security Definition	25
5.6	Relation to ACCE Model	26
6	Protocol Analyses	27
6.1	Proof of Noise Pattern XK	29
7	Discussion	35
A	Further Security Assumptions	38
A.1	Collision-Resistant Hash Functions	38
A.2	Pseudo-Random Function Security	38
A.3	AEAD security	38
A.4	The PRF-Oracle-Diffie-Hellman Assumption	38
B	Model with KCI Resistance and Randomness Reveal	39
B.1	Adversarial Model	40
B.2	Freshness regarding Corruption and Reveal of Randomness	41
B.3	Security Definition	42
C	Further Discussion of our Model	42
C.1	Restrictions of Generality	43
C.2	Strengths and Weaknesses of Matching Conversation-like Partnering	44
C.3	Mapping Stages to Round-Trips	44
C.4	Forward Secrecy within Sessions	45
C.5	On the Necessity of One Holistic Model	45
C.6	Extensions to Authentication Consideration	46
C.7	Set Randomness instead of Reveal Randomness	47
D	Correctness and Partnering Definitions in Pseudocode	47
E	Full Proofs	47
E.1	N pattern	48
E.2	NK pattern	51
E.3	NN Pattern	57

E.4	XN Pattern	59
E.5	NX Pattern	65
E.6	X pattern	71
E.7	XX pattern	76
E.8	XK pattern	84
F	Full Patterns	94

1 Introduction

Noise is a protocol framework introduced by Trevor Perrin [39] for establishing confidential channels between two parties in various application scenarios that bases on a Diffie-Hellman group, a secure key derivation function (KDF), a secure hash function, and a secure authenticated encryption with associated data scheme (AEAD). Like TLS 1.2, Noise makes use of the derived keys during channel establishment, which makes an analysis with respect to *key indistinguishability* as in traditional key exchange models infeasible. Furthermore, 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 analyze the security of patterns from the Noise framework and, since previous modeling approaches cannot be used under the aforementioned conditions, introduce the flexible ACCE model to prove fine-grained security guarantees of Noise.

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 that information has not been agreed upon by the 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, 38]) but still offers multiple protocol patterns that allow a developer to choose a protocol fulfilling their 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 share 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 via long-term DH keys, key derivation, and AEAD ciphertexts.

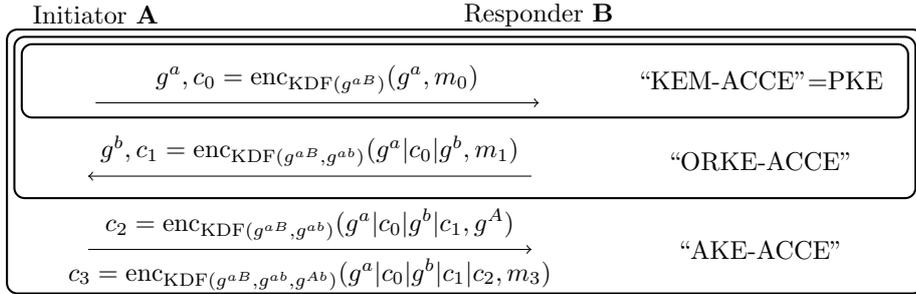


Fig. 1: The flexible structure of the Noise protocol framework, described conceptually with the XK pattern (three passes) that is based on the NK pattern (two passes), which is based on the N pattern (one pass). g^A and g^B denote the long-term public DH shares of parties **A** and **B**, g^a and g^b denote their ephemeral shares, and $\text{enc}_k(ad, m)$ 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 [25, 43] (for client to server communication), Wireguard [14, 15], Slack, Amazon AWS¹, 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

MODULARITY IN CRYPTOGRAPHY AND REAL-WORLD Definitions and analyses in cryptography usually aim to be as modular as possible such that the results are flexibly composable. In contrast, many real-world protocols are specifically designed for one purpose such that modularity – especially regarding single components of these protocols – is not necessary and maybe even undesired (e.g., due to worse performance).

For the generic (secure) composition of a key exchange protocol with a symmetric primitive (such as a symmetric channel), the key exchange protocol needs to provide key indistinguishability for the established symmetric key (among other properties; cf., Brzuska et al. [7]). However, if this symmetric key was used by the key exchange itself, it is not indistinguishable from a random key space

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, 35].

element anymore (as an adversary can simply check whether the challenged key was used). The same property needs to hold, and the same obstacle arises for multi-stage key exchange protocols: in order to allow for generic compositions of key exchange and symmetric protocols, the symmetric key must not be used by the key exchange protocol itself in order to maintain modular composition.

Since many real-world protocols (such as TLS 1.2, Quic, Signal, Noise, TLS 1.3 and others) disregard modularity (in the sense that key exchange and channel are inextricably intertwined), cryptographic analyses of these protocols chose one out of the following three bypassing approaches: 1. pausing the protocol before the key is internally used to prove key indistinguishability at that point (which still prevents generic composition results as the protocol uses the key afterwards), 2. analyzing a modified version of the protocol in which key exchange and channel are cleanly divided (which proves nothing about the actually used protocol), or 3. considering the security of the whole protocol instead of its single components by applying the ACCE model. It is important to highlight that neither of the first two approaches yields a security statement for the actually used protocol (in which e.g., a symmetric channel is initialized by the symmetric established key). Consequently, the third approach is from our point of view not only justified but also morally superior if the actual security of real-world protocols with the described lack of modularity should be assessed.

FLEXIBILITY AND GENERALIZATION FOR ACCE Originally the Authenticated and Confidential Channel Establishment (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. [32] introduced a variant of the ACCE model, allowing them to analyze protocols with unilateral authentication. In addition, Lychev et al. [36] 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.

Hence, our consideration of ACCE primitives differs from previous approaches 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. Furthermore, we drop length-hiding property [27] since we consider it not inherent in channel protocols.

After eliminating the boundary between handshake and channel, it is important to note 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 [23]. 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.

As such we follow a similar approach as the multi-stage key exchange (MSKE) model. However, since our syntax allows for no distinction between stages (note that the MSKE model obtains new keys for each stage from the protocol), we assume the considered protocols to output a *stage number* ς with every encryption and decryption. With ς , the protocol 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 generic indication to differentiate multiple stages based on our syntax³, it is essentially necessary for defining security (independent of a specific protocol) that the protocol itself outputs the stage numbers. Using these output stage numbers, one can specify for each stage which properties need to be reached by the protocol in order to achieve *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.

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 valuable security statements on channels that are built using ‘internal’

³ 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).

symmetric keys (for which composition results of the MSKE models can naturally provide no generic guarantees).⁴

CONTRIBUTIONS Our contributions can be summarized as follows:

- We generalize and flexibilize ACCE by finding its core idea and removing remnants of historic constructions and thereby propose a model to analyze channel establishment protocols with multiple stages, fulfilling different security properties. Though this model is due to its flexibility rather complex, we consider the overall generalizations useful for future analyses.
- 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. Here we give an intuition for our overall proof approach and depict the proof of pattern `XK` in full details.
- 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 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., [13, 15, 16, 18, 32, 37]). 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. [28] (which was later also used in [3, 6, 8, 9]) 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. [32] and Kohlar et al. [31] 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 [13, 15, 37]. 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 (MSKE) model by Fischlin and Günther [18] extends the Bellare-Rogaway model [2] (further extended by [16, 19]) similarly as we extend the original ACCE model (by allowing protocols to reach different

⁴ The composition theorems by Fischlin and Günther [16, 18, 19] explicitly exclude internally used keys (such that internal keys in Quic or TLS 1.3 cannot be used for generic symmetric primitives).

security properties at different stages during the execution). Due to the issue of key-usage during the handshake in Noise (as in TLS 1.2 or Signal) and further model restrictions, the multi-stage key exchange cannot be applied here.

Giesen et al. [20] 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. [36] and more recently by Chen et al. [12]. 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. 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 [14], 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. [35] 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 et al. [29, 30] 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. We note that symbolic analyses disregard the actual representation of algorithms’ 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 these primitives. Nevertheless, automatic proofs are less error-prone and better scalable which enables Kobeissi et al. [30] to apply their analysis of even more security properties (e.g., multiple variants of forward-secrecy) on far more Noise patterns than our manual approach allows.

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 further variants 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 $\{1, \dots, q\}$. 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 $x \leftarrow_{\S} S$. Let \mathcal{A} be a probabilistic algorithm, we let $y \leftarrow_{\S} \mathcal{A}(x_1, \dots)$ denote running \mathcal{A} on input (x_1, \dots) with uniformly chosen random coins, and assigning the output to y . If \mathcal{A} is a deterministic algorithm, then $y \leftarrow \mathcal{A}(x_1, \dots)$ denotes that y is computed by \mathcal{A} using (x_1, \dots) as input. By $y \leftarrow_{[r]} \mathcal{A}(x_1, \dots)$ we denote that a probabilistic algorithm \mathcal{A} is invoked deterministically by *consuming* its random coins from r (i.e., each random coin from r is used at most once). 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

Here we give the symmetric variant of the generic PRF-ODH assumption, introduced by Dowling and Paterson [15]. Our modification additionally allows to capture a “dual-PRF” like assumption necessary for the Noise Protocol Framework. The basic PRF-ODH assumption was introduced Jager et al. [27] and discussed in detail by Brendel et al. [5].

Definition 1 (Dual generic PRF-ODH Assumption). *Let G be a cyclic group of order q with generator g . Let $\text{PRF} : G \times \mathcal{M} \rightarrow \mathcal{K}$ 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 second PRF family $\text{PRF}_d : \mathcal{M} \times G \rightarrow \mathcal{K}$, by setting $\text{PRF}_d(m, e) = \text{PRF}(e, m)$. We define a security notion, **sym-lr-PRF-ODH security**, which is parameterised by: $l, r \in \{n, s, m\}$ 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 $\mathcal{G}_{\text{PRF}, G, p, \mathcal{A}}^{\text{sym-lr-PRF-ODH}}$ between a challenger \mathcal{C} and adversary \mathcal{A} .*

1. The challenger \mathcal{C} samples $u, v \leftarrow_{\S} \mathbb{Z}_p$ and provides g, g^u, g^v to the adversary \mathcal{A} .

2. If $l = m$, \mathcal{A} can issue arbitrarily many queries to oracle ODHu , and if $r = m$ and $\text{sym} = \text{Y}$ 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 $y \leftarrow \text{PRF}_\lambda(S^u, x)$ 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 $y \leftarrow \text{PRF}_\lambda(T^v, x)$ 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 $b \leftarrow_{\S} \{0, 1\}$ uniformly at random, computes $y_0 = \text{PRF}_\lambda(g^{uv}, x^*)$, and samples $y_1 \leftarrow_{\S} \{0, 1\}^\lambda$ 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 $y \leftarrow \text{PRF}_\lambda(S^u, x)$.
 - 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 $y \leftarrow \text{PRF}_\lambda(T^v, x)$.
5. At some point, \mathcal{A} outputs a guess bit $b' \in \{0, 1\}$.

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

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

We define the advantage of \mathcal{A} in breaking the dual security of PRF-ODH as:

$$\text{Adv}_{\text{PRF}, G, p, \mathcal{A}}^{\text{d-PRF-ODH}} = \max \left\{ \text{Adv}_{\text{PRF}, G, p, \mathcal{A}}^{\text{sym-lr-PRF-ODH}}, \text{Adv}_{\text{PRF}_d, G, p, \mathcal{A}}^{\text{sym-lr-PRF-ODH}} \right\}$$

Intuitively, the sym-lr-PRF-ODH assumption holds if the advantage $\text{Adv}_{\text{PRF}, G, p, \mathcal{A}}^{\text{sym-lr-PRF-ODH}}$ of any PPT adversary \mathcal{A} is negligible. For conciseness in the advantage statements, we omit the d-PRF-ODH , and instead use sym-lr to specify which PRF-ODH assumption we use. Further used variants of the assumption are in Appendix A.

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 (in addition to confidentiality) 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 executions' random coins.

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 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.

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 [39].

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 future⁵ – 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.

Furthermore, we do not consider the associated data input on sending and receiving payload after the handshake. As our syntax intentionally makes no difference between handshake and channel, we cannot consider this additional feature of the Noise channel, as it is not provided during the handshake. Finally, we assume the protocols to output information on the current level of security (which we explain in more details below).

⁵ <https://moderncrypto.org/mail-archive/noise/2018/001495.html>

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 patterns, the single letter of the pattern name indicates whether the initiator’s long-term public key is not defined (**N**), trans(X)mitted during the handshake (**X**), or known 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 key pair (of 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 plain 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 (Identify) DH share in plaintext (i.e., **I_**).

At the left margin of Figure 1, we depict how the Noise patterns’ algorithms are invoked for party **A** (matching our generic syntax, formally defined in Section 5).

The handshake of a Noise pattern always starts with the initialization of the local state *st* (via `Init()`). This local state contains:

1. ρ : a boolean that indicates the session’s role (initiator/responder),
2. **pattern**: the pattern name,
3. (X, g^X) : the session owner’s long-term DH exponent and DH share (optional),
4. g^Y : the intended partner’s long-term public DH share (optional),
5. (x, g^x) : the session’s ephemeral DH exponent and DH share (optional),
6. g^y : the peer’s ephemeral public DH share (optional),
7. *ck*: the chaining key,
8. *h*: the hash variable,
9. *k* or k_i, k_r : the encryption key(s), and
10. *n* or n_i, n_r : the nonce(s) for encryption.

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).

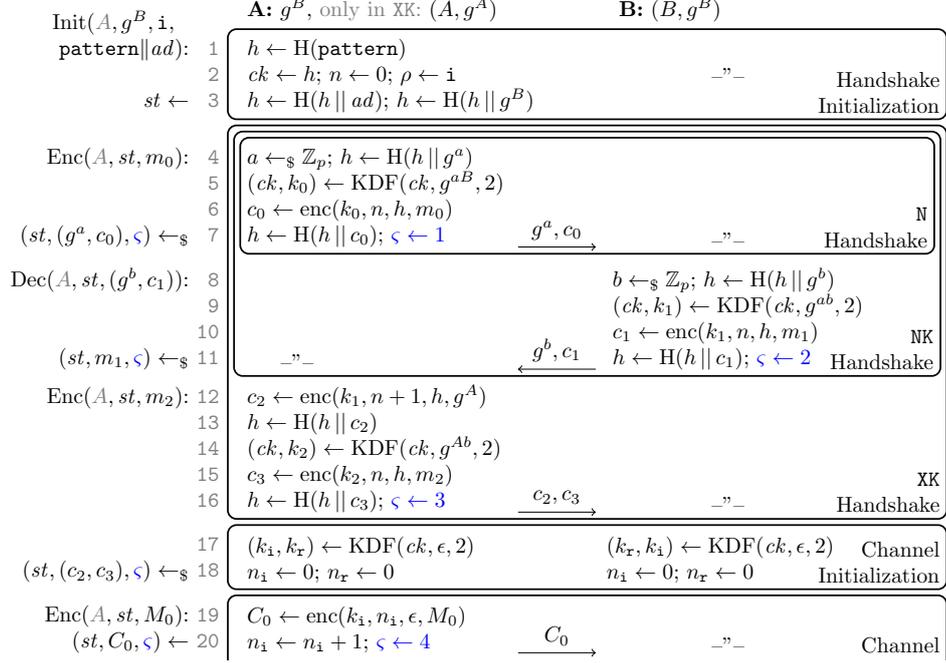


Fig. 2: Fully specified N,NK and XK patterns. m_i are payload messages sent *during* the handshake; M_i are payload messages sent *after* the handshake; ad is associated data with which the handshake is initiated; “--” denotes that the respective operations for receipt are processed (e.g., $\text{dec}(c, ..)$ for $c \leftarrow \text{enc}(..)$). Handshake initialization, channel initialization, and channel are part of all patterns. Algorithm invocations and return values in the left column depict the interaction of party **A** with the protocol (showing the protocol’s syntax defined in Section 5). Blue marked parts are not specified by Noise but are required for our analysis (thus we assume them to be part of the protocol).

For each encryption (via Enc) during the handshake (i.e., before all desired security properties are reached), 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 4,8). In case (b), the sender’s long-term DH share is encrypted under the current key k and the resulting ciphertext is hashed

into h and sent to the partner (lines 12-13).⁶ If (c) a DH secret is computed, the current ck together with this DH secret are given as input to an invocation of the KDF (lines 5,9,14).

For each encryption during the handshake 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 .⁷ This ciphertext is sent to the partner and is also hashed into h . The current value of h is associated data for every encryption (lines 6-7,10-11,15-16).

After all handshake ciphertexts are processed, the channel is initialized with a symmetric key for each communication direction, derived by invoking the KDF on the current chaining key ck (lines 17-18). In one-message patterns such as N, payload can however only be sent from initiator to receiver.

Please note that we assume the protocol to additionally output information on the current payload transmission's level of security (represented by integer ς ; lines 7,11,16,20). We proposed to add this feature to the Noise specification via the mailing list (as it could be useful to upper layer protocols). In Section 5, we describe why this feature is necessary for a security analysis in a generic model and in Section 6 we explain how this *stage counter* is naturally derived for the Noise patterns.

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 Replay Attacks, State Reveals, and Their Relation

In our model, presented in Section 5, we allow adversaries to reveal the secret local states of session participants. Since this slightly raises the complexity of the model – as it induces a more careful treatment of *trivial* attacks – we contextualize the meaning and explain the importance of this adversarial power (in order to justify the increased complexity), and we give an intuition for relations between state reveals and replay attacks on a high level (in order support comprehensibility of the model). This is aimed to give some initial motivation on how we define the model formally in the next section.

Replay Attacks in Noise Replay attacks are an inevitable issue for early communication in many protocols – among them, many Noise patterns (e.g., patterns N,

⁶ For patterns in which the long-term DH share is sent in plaintext, this DH share is directly hashed into h instead.

⁷ Note that we use the algorithm Enc generically for sending information to the session partner. Confidentiality of payload is thereby not necessarily reached (see Section 5.1).

NK, and XK, cf., Figure 2). When assuming long-term keys of parties to be constant (and not variable; cf., [23]), the first ciphertext in a session, sent from an initiator instance π_i^s of a party i to a responder instance $\pi_j^{t_1}$ of a party j , can be replayed to all other (responder) instances $\pi_j^{t_2}, \dots, \pi_j^{t_n}$ of party j . As long as instances of party j are not synchronized, they will not detect this replay attack (since the ciphertext is valid for all of them). Hence, they will all accept and process this ciphertext and reply with individual (valid) ciphertexts as responses.

We observe three conditions that allow for replay attacks and that are true for seven out of the 15 standard patterns in Noise (cf., rows in Table 1 with $\text{rt} = 0$): (1) parties’ long-term keys are static, (2) first ciphertexts in sessions from initiator to responder contain (confidential) payload already, and (3) there exists no (specified) synchronization mechanism among instances of a party. As a result, such ciphertexts in these patterns are inevitably potential subject to replay attacks.

Importance of State Reveal In general – independent of replay attacks – local states of instances contain crucial session secrets. Since the primitive that we consider in this work depicts not only the initialization of a session but the session itself (in contrast to, e.g., authenticated key exchange), these considered local secrets are stored and used until a session is terminated. For settings with long session duration (e.g., IoT networks), it is reasonable to assume that adversaries gain access to some instances’ local secrets during the session lifetime. As a result, a realistic model should capture this adversarial power by allowing state reveals in the security game.

If state reveals were not allowed, protocols that store valuable secrets unnecessarily in the local state (e.g., own or partners’ long-term secrets) would be declared secure even though this is intuitively insecure.

State Secrets under Replay Attacks In the following we describe an attack against an intuitively insecure protocol that would formally be declared “secure” in a model without state reveal. Consequently, we argue that such models are unsuitable for assessing *security*.

1. An initiator instance π_i^s of party i sends the first ciphertext c_0 in a session directed to a responder instance of party j , containing an ephemeral public encryption key pk^* , and stores the respective secret key sk^* in its local state. This ciphertext is not protected against replay attacks (for the reasons given above).
2. This ciphertext c_0 is forwarded to multiple responder instances $\pi_j^{t_1}, \dots, \pi_j^{t_n}$ of party j .
3. Each responder instance $\pi_j^{t_l}, l \in [n]$ encrypts its individual, independent, confidential reply payload under pk^* in a ciphertext \tilde{c}^l . We note that this payload is encrypted forward-securely as (sk^*, pk^*) are independent of i ’s and j ’s long-term keys. Each instance $\pi_j^{t_l}$ sends a replay-resistant response c_1^l back to instance π_i^s , such that c_1^l contains \tilde{c}^l .

4. Instance π_i^s will only receive and process one reply $c_1^{t^*}$ from an instance $\pi_j^{t^*}$ and will continue a long-lived session with it. Instance $\pi_j^{t^*}$ encrypts all further payload to π_i^s under pk^* such that sk^* remains in π_i^s 's local state until the session terminates.
5. All remaining involved sessions of party j will eventually terminate due to a timeout.
6. If an attacker obtains the local state of π_i^s before its session with $\pi_j^{t^*}$ terminates, it will learn all confidential replies, encrypted in ciphertexts $\tilde{c}^1, \dots, \tilde{c}^n$ (in addition to the entire payload from $\pi_j^{t^*}$).

A model without state reveal declares all replies from instances $\pi_j^{t_1}, \dots, \pi_j^{t_n}$ “secure” even though their security crucially relies on the secrecy of π_i^s 's local state. Especially since countermeasures are trivial and highly efficient⁸, we consider this example protocol insecure. This intuition matches the initial idea of the (key) reveal query in the Bellare-Rogaway model [2]: “Compromise of a session key should have minimal consequences” such that it should not “leak information about other (as yet uncompromised) session keys”. Since we abstractly consider all local secrets of an instance combined as a generic state st , this condition should hold accordingly for its reveal. For protocols with early communication (allowing for replay attacks), this condition can, however, only be met as soon as a session has exactly two participants (i.e., the session continues with exactly one responder).

Depiction in the Model The essence of the above attack is not that messages are in danger due to replay attacks, but rather that secrets established during replay attacks may affect multiple instances. This effect must be reduced by secure protocols as far and as soon as possible. In our model, stage counters rp^i, rp^r indicate how soon initiators and responders have local states that are independent of other instances' secrets (see Section 5.3). Thereby, we define reveals of their states harmless (for other instances) in case they are conducted after these stages are reached. Even though this independence may not be reached immediately (due to replay attacks in early communication), our model transparently indicates with these counters, how soon it is reached by protocols (see Table 1).

In our proof we explicitly emphasize the game hops in which it becomes clear that secrets are independent of others that are in revealable states of other instances.

5 Flexible ACCE Framework

The original ACCE model [27] and our generalization focus on the definition of *authentication* and *confidentiality* of messages, transmitted via a communication

⁸ E.g., $\pi_j^{t_1}, \dots, \pi_j^{t_n}$ encrypt individual random symmetric keys k^1, \dots, k^l under pk^* and encrypt their content under these symmetric keys respectively, then π_i^s can erase sk^* quickly such that its state is free of session-overlapping secrets.

protocol (*channel establishment*). However in [27], traditional security goals like authentication and forward-secrecy are required to be reached before the actual channel is established.

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 define fACCE security.

5.1 fACCE Primitive Description

Below we define the *flexible ACCE* primitive. Intuitively, fACCE 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.⁹ The special ‘security level’-output ς of encryption and decryption signals which security properties are reached by the current algorithm invocation (e.g., to a higher level application).

Definition 2 (Flexible ACCE). *A flexible ACCE protocol fACCE is a tuple of algorithms $\text{fACCE} = (\text{KGen}, \text{Init}, \text{Enc}, \text{Dec})$ 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:*

- $\text{KGen} \rightarrow_{\S} (sk, pk)$ generates a long-term key pair where $sk \in \mathcal{SK}, pk \in \mathcal{PK}$.
- $\text{Init}(sk, ppk, \rho, ad) \rightarrow_{\S} st$ 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 intended session partner, $\rho \in \{\mathbf{i}, \mathbf{r}\}$ is the session’s role (i.e., initiator or responder), ad is data associated with this session, and $sk \in \mathcal{SK} \cup \{\perp\}, ppk \in \mathcal{PK} \cup \{\perp\}, ad \in \{0, 1\}^*, st \in \mathcal{ST}$.
- $\text{Enc}(sk, st, m) \rightarrow_{\S} (st', c, \varsigma)$ 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 $sk \in \mathcal{SK} \cup \{\perp\}, st, st' \in \mathcal{ST}, m, c \in \{0, 1\}^*, \varsigma \in \mathbb{N}$.
- $\text{Dec}(sk, st, c) \rightarrow_{\S} (st', m, \varsigma)$ 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 $sk \in \mathcal{SK} \cup \{\perp\}, st \in \mathcal{ST}, st' \in \mathcal{ST} \cup \{\perp\}, m, c \in \{0, 1\}^*, \varsigma \in \mathbb{N}$. If $st' = \perp$ is output, then this denotes a rejection of this ciphertext.

We define as a convention that for output stage numbers $\varsigma = 0$, no security properties (in particular, no confidentiality) for the respectively transmitted payload has yet been reached.¹⁰ Further we assume that the output stage numbers monotonically increase during a session (which is not a restriction).

⁹ 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.

¹⁰ It is important to note that the first stage may not necessarily be 0, e.g. 0-RTT protocols that achieve confidentiality with the first message.

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., protocols enforcing correct message order, and aborting for message omissions). In Appendix C.2 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 from the established channel, were equally sent to this channel by the partner. In Appendix D Figure 7 we define correctness in a pseudocode notation to provide more clarity.

Definition 3 (Correctness of fACCE). *An fACCE protocol is correct if, for any two key pairs $(sk_i, pk_i), (sk_r, pk_r)$ output from KGen or set to (\perp, \perp) respectively, their session states $\text{Init}(sk_i, pk_i, i, ad) \rightarrow_{\S} st_i, \text{Init}(sk_r, pk_r, r, ad) \rightarrow_{\S} st_r$ with $ad \in \{0, 1\}^*$, and message-stage-ciphertext transcripts $MSC_\rho, MSC_{\bar{\rho}} \leftarrow \epsilon$, it holds for all sequences of operations $((op^0, \rho^0, m^0), \dots, (op^n, \rho^n, m^n))$ (for all $0 \leq l \leq n$ with $op^l \in \{e, d\}, \rho^l \in \{i, r\}, m^l \in \{0, 1\}^*$) that are executed as follows:*

- if $op^l = e$, invoke $\text{Enc}(sk_{\rho^l}, st_{\rho^l}, m^l) \rightarrow_{\S} (st_{\rho^l}, c^l, \zeta^l)$ and update $MSC_\rho \leftarrow MSC_\rho \parallel (m^l, \zeta^l, c^l)$, or
- if $op^l = d$, invoke $\text{Dec}(sk_{\rho^l}, st_{\rho^l}, c^l) \rightarrow_{\S} (st_{\rho^l}, m_*^l, \zeta_*^l)$ on $(m_\circ^l, \zeta_\circ^l, c^l) \parallel MSC_{\bar{\rho}} \leftarrow MSC_{\bar{\rho}}$ and update it accordingly,

that if $m_^l \neq \perp$, then encrypted and decrypted messages and stage outputs equal $m_*^l = m_\circ^l, \zeta_*^l = \zeta_\circ^l$, and that stage outputs increase monotonically ($\forall l^* < l$ with $op^l = op^{l^*} = e$ and $\rho^{l^*} = \rho^l$ it holds that $\zeta^{l^*} \leq \zeta_\circ^l$).*

5.2 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 a consequence, one can specify for each stage which properties need to be reached by the protocol in order to achieve *security*. Since one security property may not be reached in an early stage (thus the adversary could trivially attack communication in this stage) and later stages may reach this security property, 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. We note that due to allowing secure and insecure stages within the same session, dependencies between messages may leak information to an attacker.

We consider a set of n_P parties each (potentially) maintaining a long-term key pair $\{(sk_1, pk_1), \dots, (sk_{n_P}, pk_{n_P})\}, (sk_i, pk_i) \in \mathcal{SK} \times \mathcal{PK}$. 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, that indicates whether sk_i was exposed to the adversary (via `OCorrupt`, see Section 5.4).

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.

- $\rho \in \{\mathbf{i}, \mathbf{r}\}$: 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 variables¹¹.
- $rand \in \{0, 1\}^*$: Any random coins used by π_i^s ’s protocol execution.
- (b_1, b_2, b_3, \dots) : A vector of challenge bits the adversary is to guess (one bit for each stage).
- $(fr_1, fr_2, fr_3, \dots)$: 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 random coins $rand$ have been revealed to the adversary.¹²

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

Furthermore a set of ciphertexts $Rpl \leftarrow \emptyset$ is maintained in the security game, that are declared to initiate a non-fresh (replayed) 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 (`OEnc`, `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 once¹³. This definition considers the asynchronous nature of the established channel, leading to a *matching conversation*-like partnering definition for `FACCE`.

¹¹ See Subsection 3.1 Noise’s state definition.

¹² Please note that this variable is only necessary for the full model, described in Appendix B.

¹³ 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 4 (Honest Partner). π_j^t is an honest partner of π_i^s if all initial variables match ($\pi_i^s.pid = j$, $\pi_j^t.pid = i$, $\pi_i^s.\rho \neq \pi_j^t.\rho$, $\pi_i^s.ad = \pi_j^t.ad$) and the received transcripts are a prefix of the partner’s sent transcripts, respectively, where at least one them is not empty (i.e., for $a = |\pi_j^t.T_d|$, $b = |\pi_i^s.T_d|$ such that $a > 0$ if $\pi_i^s.\rho = \mathbf{i}$ and $b > 0$ if $\pi_i^s.\rho = \mathbf{r}$ 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 $\text{Partner}(i, s)$ 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 4, our partnering notion requires that, after decrypting once, a session must have no more than one honest partner. Thereby partnering necessarily becomes a 1-to-1 relation as soon as the initiator received once from the responder.

In Appendix C.2 we discuss the advantages and drawbacks of our partnering notions as well as extensions to consider unreliable networks.

5.3 Flexible Security Notion

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

In order to keep the presentation of the model comprehensible, we do not consider certain 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 [18] 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 suffices¹⁴. Since most real-world protocols adhere to this, our security definition is indexed by ten integers, called counters, ($\mathbf{au}^i, \mathbf{au}^r, \mathbf{kc}^i, \mathbf{kc}^r, \mathbf{fs}, \mathbf{eck}, \mathbf{rl}^i, \mathbf{rl}^r, \mathbf{rp}^i, \mathbf{rp}^r$) 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 $\rho \in \{\mathbf{i}, \mathbf{r}\}$ (for initiator and responder respectively). One can think of each counter as a reference ‘rung’ on the ‘ladder’ of

¹⁴ We make use of this but one can easily extend our model to allow analyses of protocols with decreasing security properties.

stages from which on the specified security property is achieved by the respectively analyzed protocol. Thus, as soon as the protocol outputs a certain stage that equals a counter (the protocol says that it reached the indicated ‘rung’ on the ‘ladder’), all messages that are transmitted thereafter (including the message just encrypted or decrypted) reach the corresponding security property. Please note that some security properties, such as authentication, develop their effect in two steps (see the trivial and real attacks in the description of oracle ODec in Section 5.4). As mentioned above, we only explain the counters that are necessary for the full model ($\mathbf{kc}^i, \mathbf{kc}^r, \mathbf{eck}, \mathbf{r1}^i, \mathbf{r1}^r$) in Appendix B. We describe the (remaining) counters below:

1. \mathbf{au}^ρ defines the stage required for ρ to be authenticated. 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 \mathbf{au}^ρ . Note that, since our partnering notion considers FIFO channels, thereby also ciphertexts (and the order among them) are authenticated that were sent and received before \mathbf{au}^ρ was reached.
2. \mathbf{fs} defines the stage from which forward-secrecy (with respect to both session participants’ long-term secrets) is reached. It is hard, for a stage $\varsigma \geq \mathbf{fs}$, to break the confidentiality of ciphertexts, even if both parties were corrupted (*unless* one of the session participants’ random coins were revealed to the adversary; see the full model).¹⁵
3. \mathbf{rp}^ρ defines the stage from which a fully revealed session state of ρ cannot be used to replay and reestablish the session. This means, for a session for which stage $\varsigma \geq \mathbf{rp}^\rho$ was reached, a revealed session state must not contain secrets that affect the communication’s security of any non-partnered sessions (especially of other receivers). The second condition of our partnering notion (cf., Definition 4) divides partners after replay attacks occur (i.e., marks them unpartnered thereby). Hence, protocols must diverge session state(s) of previous partners in case of such replay attacks, if partnering is used to control (and forbid) session state reveals. Only replayed first ciphertext(s) from an initiator to a responder do not to divide partners according to our partnering notion. These latter ‘unpreventably’ replayed ciphertexts establish secrets in the local session state of the sender and receivers such that these local states might be dependent (among sender and potentially multiple receivers of a replayed ciphertext). 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.¹⁶

¹⁵ 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.7.

¹⁶ One could extend the precision regarding replayability by further considering properties such as post-compromise security, whether the long-term key was already used,

We remark that our partnering notion already defines session participants unpartnered for all but one type of replay attacks: if ciphertexts, sent by an initiator that has already received a ciphertext once, or sent by a responder, are replayed, the respective receiver is defined to have no honest partner. In a security game in which state reveals are defined to be harmless for unpartnered sessions (which is the case for our model), this induces that such replay attacks force the protocol to diverge respective receivers' session states from their previous partners' session states. As a consequence, only replays of ciphertexts, sent by an initiator to (multiple) responder(s) without any reply from the latter, must be considered harmful in our 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., [23]) 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 (other) 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.

Besides the explained prevention of replay attacks due to our partnering notion, ciphertexts that are transmitted before a stage $\varsigma > 0$ is output are (as also explained above) authenticated as soon as authentication is reached in a later stage. Apart from this, no security guarantees are required for ciphertexts transmitted under $\varsigma = 0$.

If a property is never reached in the specified protocol, then the respective counter is set to ∞ (e.g., for protocol with unauthenticated initiators, $\text{au}^1 = \infty$).

5.4 Adversarial Model

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

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 [42]), we treat these two properties similar to the original AEAD notion of Rogaway [41]: the game maintains a win flag (to indicate whether the adversary broke authenticity or

or whether asymmetric cryptography is used (which we consider not useful in this setting).

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 win $\leftarrow 1$ or output the correct challenge bit $\pi_i^s.b_\zeta$ of a specific session stage ζ at the end of the game.

- $\text{OInit}(i, s, j, \rho, ad)$ initializes a session π_i^s (if not yet initialized) of party i to be partnered with party j , invoking $\text{fACCE.Init}(sk_i, pk_j, \rho, ad) \rightarrow_{[\pi_i^s.rand]} \pi_i^s.st$ 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 $\text{Fresh}_{\text{fs}}()$ (see Figure 3).
- $\text{OEnc}(i, s, m_0, m_1)$ triggers the encryption of message m_b for $b = \pi_i^s.b_\zeta$ by invoking $\text{Enc}(sk_i, \pi_i^s.st, m_b) \rightarrow_{[\pi_i^s.rand]} (st', c, \zeta)$ for an initialized π_i^s if $|m_0| = |m_1|$ and for $\zeta = 0$ (i.e., confidentiality is not yet achieved) it must hold that $m_0 = m_1$ as the challenge bit would otherwise be trivially leaked. It updates the session specific variables $\pi_i^s.st \leftarrow st'$, returns (c, ζ) to the adversary, and appends c to $\pi_i^s.T_e$ if $c \neq \perp$.
- $\text{ODec}(i, s, c)$ triggers invocation of $\text{Dec}(sk_i, \pi_i^s.st, c) \rightarrow_{[\pi_i^s.rand]} (st', m, \zeta)$ for an initialized π_i^s and returns (m, ζ) if π_i^s has no honest partner, or returns ζ 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:

fs: Since decryption can change the honesty of partners, the freshness flags are updated regarding corruptions by invoking $\text{Fresh}_{\text{fs}}()$ (see Figure 3).

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 ($\pi_i^s.fr_{\zeta^*} \leftarrow 0$ for all $\zeta \leq \zeta^*$), since this is a KCI attack.¹⁷
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., $\zeta < \text{au}^{\pi_i^s.\bar{\rho}}$), then all following stages are marked un-fresh until authentication will be reached ($\pi_i^s.fr_{\zeta^*} \leftarrow 0$ for all $\zeta \leq \zeta^* < \text{au}^{\pi_i^s.\bar{\rho}}$), since this is a (temporary) trivial impersonation of the partner towards π_i^s .¹⁸
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., $\zeta \leq \text{au}^{\pi_i^s.\bar{\rho}}$), then all

¹⁷ Please note that in the reduced model (in contrast to the full model in Appendix B), resistance against KCI attacks is not required.

¹⁸ 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.

following stages are marked un-fresh ($\pi_i^s.fr_{\zeta^*} \leftarrow 0$ for all $\zeta \leq \zeta^*$), 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 $\text{win} \leftarrow 1$ 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 ($\pi_i^s.fr_{\zeta} = 1$), and

1. this is the first authenticated ciphertext ($\zeta = \text{au}^{\pi_i^s.\bar{\rho}}$), and neither party i nor the intended partner are corrupted ($\text{corr}_i = \text{corr}_{\pi_i^s.pid} = 0$), or
 2. this is a later authenticated ciphertext ($\zeta > \text{au}^{\pi_i^s.\bar{\rho}}$) and party i is not corrupted ($\text{corr}_i = 0$) as this would otherwise be a KCI attack.
- **OCorrupt**(i) $\rightarrow sk_i$ outputs the long-term secret key sk_i of party i , sets $\text{corr}_i \leftarrow 1$, and updates the freshness flags by invoking $\text{Fresh}_{\text{fs}}()$.
 - **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 determines this session's challenge bits, since the state contains any used session keys¹⁹. Hence $\pi_i^s.fr_{\zeta^*} \leftarrow 0$ is set for all stages ζ^* .
 - Similarly, sufficient information is leaked to determine challenge bits embedded in ciphertexts to and from *all* honest partners π_j^t (and to impersonate π_i^s towards them). As such, $\pi_j^t.fr_{\zeta^*} \leftarrow 0$ is set for all stages ζ^* 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 ($\zeta < \text{rp}^{\pi_i^s.\rho}$ where ζ 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 $c \leftarrow \pi_i^s.Te[0]$ if $\pi_i^s.\rho = \mathbf{i}$ or $c \leftarrow \pi_i^s.Td[0]$ if $\pi_i^s.\rho = \mathbf{r}$ (such that all sessions starting with this ciphertext are also marked non-fresh)²⁰.

Freshness regarding Corruptions and Replays The definition of forward-secrecy, based on counter **fs**, is straight forward: if either the own long-term secrets or the intended partner's long-term secrets were corrupted (i.e., $\text{corr}_i = 1 \vee$

¹⁹ 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.

²⁰ 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. [24]). 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.

$corr_{\pi_i^s.pid} = 1$), then only stages that provide forward-secrecy are marked fresh for the respective session (i.e., $\pi_i^s.fr_{\zeta^*} \leftarrow 0$ for all $\zeta^* < \mathbf{fs}$). For sessions started with a ciphertext marked in set Rpl (i.e., initiating insecure communication due to the reveal of a replayable session), all stages are marked insecure. We formally define these properties via function $\text{Fresh}_{\mathbf{fs}}()$ (see Figure 3) because this simplifies the definition of freshness in the full model (via $\text{Fresh}_{full}()$) in which also sessions' random coins can be revealed.

```

Freshfs():
For all  $i \in [n_P]$ , for all  $s \in [n_S]$ :
   $\mathbf{ctr} \leftarrow \min(\zeta^* : \pi_i^s.fr_{\zeta^*} = 1)$ 
  If  $corr_i = 1 \vee corr_{\pi_i^s.pid} = 1$ :
     $\mathbf{ctr} \leftarrow \max(\mathbf{ctr}, \mathbf{fs})$ 
  If  $\pi_i^s.Te[0] \in Rpl \wedge \pi_i^s.\rho = \mathbf{i}$ :
     $\mathbf{ctr} \leftarrow \infty$ 
  If  $\pi_i^s.Ta[0] \in Rpl \wedge \pi_i^s.\rho = \mathbf{r}$ :
     $\mathbf{ctr} \leftarrow \infty$ 
   $\pi_i^s.fr_{\zeta^*} \leftarrow 0$  for all  $\zeta^* < \mathbf{ctr}$ 

```

Fig. 3: Function for updating freshness flags after each oracle invocation, considering long-term secrets' corruption (w.r.t. forward-secrecy) and full state reveals (w.r.t. replay attacks). The freshness flags up to (and excluding) the first secure stage are reset (e.g., for corrupted long-term keys, all stages in affected sessions are reset until forward-secrecy is reached). $\text{Fresh}_{full}()$ that defines freshness also under the reveal of random coins can be found in Figure 5.

5.5 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, ζ, b') .

We can now turn to defining (in-)security of a fACCE protocol.

Definition 5 (Advantage in Breaking Flexible ACCE). *An adversary \mathcal{A} breaks a flexible ACCE protocol fACCE with authentication stages $(\mathbf{au}^i, \mathbf{au}^r)$, forward-secrecy stage \mathbf{fs} , and replayability resistance stages $(\mathbf{rp}^i, \mathbf{rp}^r)$, when \mathcal{A} terminates and outputs (i, s, ζ, b') , if there either exists a session π_i^s such that $\pi_i^s.b_\zeta = b'$, and $\pi_i^s.fr_\zeta = 1$ (which we subsume as event **guess**), or $\text{win} = 1$. We define the advantage of an adversary \mathcal{A} breaking a flexible ACCE protocol fACCE as $\text{Adv}_{\mathcal{A}}^{\text{fACCE}} = (2 \cdot \Pr[\text{guess}] - 1) + \Pr[\text{win} = 1]$.*

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

Necessity of 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 Appendix C.5 we explain why this approach would produce more complexity, less comprehensibility, and is partially impossible.

5.6 Relation to ACCE Model

Our aim was specifically not to adapt the ACCE model by Jager et al. [27, 28] so far that it is “generic enough” but rather to think about the concept of *authenticated and confidential channel establishment* from scratch. Nevertheless, both models are partially comparable.

Due to differences in syntax and correctness, neither of both security definitions formally implies the other one. However, when ignoring the associated data field in the fACCE initialization algorithm, then for protocols that 1. allow for an alternating “handshake phase”, 2. use symmetric keys for channel protection, and 3. achieve correctness with respect to message transmission (and not only w.r.t. symmetric key equality) ACCE security and fACCE security for $\text{au}^i = \text{kc}^r = x$, $\text{au}^r = \text{kc}^i = y$, $\text{fs} = \text{rp}^i = \text{rp}^r = 3$, $\text{eck} = \text{r1}^i = \text{r1}^r = \infty$ with $x, y \in \{1, 2\}$ morally depict equivalent security requirements in case stage x is output for the last “handshake ciphertext” from initiator to responder, stage y is output for the last “handshake ciphertext” from responder to initiator, and stage 3 is output for all ciphertexts after the handshake.

For this statement it is important to note that authenticated encryption (on which our channel notion bases) and length-hiding authenticated encryption with associated data (on which the original ACCE model’s channel notion bases) can trivially be built from another.²¹

As an immediate reduction between ACCE security and fACCE security cannot be achieved due to the listed differences, we leave a formal treatment of these constraints and a potentially resulting prove on the relation between ACCE security and fACCE security as an open question for future work.

Restrictions of our Model’s Generality Even though our model should be independent of a specific protocol and as natural as possible, we simplified parts of it

²¹ AE is implied by lhAEAD and lhAEAD can be built from AE by concatenating (with an appropriate encoding) the associated data to the plaintext before encryption and comparing it with the associated data at decryption before removing it from the plaintext again. In order to achieve the length hiding property, one simply needs to add padding to the plaintext (in order to keep the length parameter meaningful, it would be increased by the length of the associated data). Please note that this construction is not practical but only aimed to illustrate the relations.

for clarity and made choices (if there were equivalent options) in favor of Noise. In Appendix C.1 we list these choices and simplifications for transparency.

6 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. We focus on proving these two protocol “families” to demonstrate how our analysis can capture the wide variety of security properties that we show in Table 1, while also simplifying our approach by the re-use of our proof strategies. We give a detailed look of the proofs of Noise Pattern XK in the reduced model here and extend these proofs, considering further security properties in the full model, together with the proofs for the remaining mentioned patterns in Appendix E. We present the analysis of Noise Pattern XK here as it comprehensibly provides an idea of the general proof structure and shows how Noise patterns can be built upon another. As the handshake of XK extends NK’s handshake, which in turn extends the handshake of N by a half round-trip respectively, each extension also results in further security properties (see Figure 2 and Table 1).

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.

We determine which OCorrupt (and ORevealRandomness in the full model) queries cannot have been issued such that the attacked stage is still ‘fresh’ (as the adversary would otherwise be unsuccessful). Thus, each case has some queries that have not been issued to the session π_i^s and its partner session (where π_i^s either accepted the forged ciphertext, or the adversary output (i, s, ς, b')). In both cases we use a tailored PRF-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 assumption(s) 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: 1. The adversary is unable to forge a ciphertext to the session π_i^s , or 2. 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 Here we define the exact modeled security via the stage counters $(\mathbf{au}^i, \mathbf{au}^r, \mathbf{fs}, \mathbf{rp}^i, \mathbf{rp}^r)$, $(\mathbf{kc}^i, \mathbf{kc}^r, \mathbf{eck}, \mathbf{rl}^i, \mathbf{rl}^r)$ ²², used in our theorems of each proof. We also explain how they relate to the round-trips in the protocol execution of the respective Noise pattern (we discuss generic mapping among stage counters and round-trips in Appendix C.3). 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. In the case that the initiator’s first ciphertext provides no confidentiality, the stage output is 0 (see column $rt = 0.5$ in Table 1) but the reply by the responder continues with $\varsigma = 2$.²³

The counters/round-trips for authentication and KCI resistance $(\mathbf{au}^\rho, \mathbf{kc}^\rho)$ are directly lifted from the Noise specification [39]. As the definition of the remaining security properties deviate from the specification (or are not specified therein), the theorems’ stage counters are defined as the first round-trips and stages that achieve the respective goals. Regarding forward-secrecy, the Noise specification differentiates among role dependent weak and strong variants of long-term secrets’ corruptions. However, 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 \mathbf{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

²² The latter are only relevant for the proofs in the full model.

²³ Thus the first encryption of the initiator and the first decryption of the responder output 1 for confidential payload and 0 for non-confidential payload, 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. 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 (except for first half non-confidential round-trips).

is allowed to reveal the local state, our definition of replay attack resistance goes beyond others in the literature (e.g., [19]) 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).

	rt	au ⁱ	au ^r	fs	rp ⁱ	rp ^r	kc ⁱ	kc ^r	eck	rl ⁱ	rl ^r
N*	0	∞	∞	∞	∞	∞	∞	∞	∞	1	∞
X*	0	1	∞	∞	∞	∞	∞	∞	1	1	∞
K	0	1	∞	∞	∞	∞	∞	∞	1	1	∞
NN*	0.5	∞	∞	2	2	0	∞	∞	∞	∞	∞
NK*	0	∞	2	2	2	2	∞	2	∞	1	∞
NX*	0.5	∞	2	2	2	0	∞	2	∞	2	∞
XN*	0.5	3	∞	2	2	0	3	∞	∞	∞	3
XK*	0	3	2	2	2	2	3	2	∞	1	3
XX*	0.5	3	2	2	2	0	3	2	∞	2	3
KN	0.5	3	∞	2	2	0	3	∞	∞	∞	2
KK	0	1	2	2	2	2	3	2	1	1	2
KX	0.5	3	2	2	2	0	3	2	∞	2	2
IN	0.5	3	∞	2	2	0	3	∞	∞	∞	2
IK	0	1	2	2	2	2	3	2	1	1	2
IX	0.5	3	2	2	2	0	3	2	∞	2	2

Table 1: Stages at which the respective security properties are reached. Stage x is reached (and thus returned by the protocol via output ς) at round-trip $RT(x) = x/2$ (for $RT(x) < rt$ no property is reached). The right half of columns depicts the counters for security properties that are only considered in the full model. \mathbf{au}^p , \mathbf{kc}^p were extracted from Noise’s specification [39]; \mathbf{fs} , \mathbf{rp}^p are related to their definition in the specification (but adapted to our model). \mathbf{rl}^p , \mathbf{eck} were defined purely with respect to the model. We give proofs for the patterns marked with a *.

6.1 Proof of Noise Pattern XK

Theorem 1. *Noise protocol XK (as in Figure 2) is an fACCE-secure protocol with authentication levels $\mathbf{au} = (3, 2)$, forward-secrecy $\mathbf{fs} = 2$, and replay resistance $\mathbf{rp} = (2, 2)$. For an adversary \mathcal{A} against the flexible ACCE security game (defined in Section 5) one can efficiently define adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of \mathbb{H} , $\mathcal{B}_{\text{PRF-ODH}}$ against the PRF-ODH assumptions ms-PRF-ODH, sn-PRF-ODH and sym-ms-PRF-ODH with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the*

PRF security of KDF with:

$$\begin{aligned}
\text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE}} &\leq 3 \cdot \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S \cdot \left(\text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} \right. \\
&\quad \left. + \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right) + n_P^2 n_S^2 \cdot \left(\text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} \right) \\
&\quad + n_P^2 n_S^2 \cdot \left(\max \left\{ \left(3 \cdot \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + 4 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right), \right. \right. \\
&\quad \left. \left. \left(2 \cdot \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + 3 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sn-PRF-ODH}} \right) \right\} \right).
\end{aligned}$$

Proof. We give below the proof of Noise Pattern XK. We split our analysis into three cases, depending on *how* the adversary can win the experiment. For the first two cases, the adversary causes $\text{win} \leftarrow 1$ if the received ciphertext was not sent by a session of the intended partner, but was successfully decrypted in an authenticated stage by either an initiator (**Case A**) or a responder (**Case B**) session. For **Case A** \mathcal{A} cannot have issued a $\text{OCorrupt}(\pi_i^s, \text{pid})$ query because breaking authentication of a corrupted peer is a trivial attack (as $\text{au}^r = 2$). Similarly, for **Case B** \mathcal{A} cannot have issued a $\text{OCorrupt}(\pi_i^s, \text{pid})$ query as $\text{au}^i = 3$. Next we focus on an adversary attempting to guess the challenge bit b for any fresh session (**Case C**). **Case C** is further separated into two subcases, depending on the combination of allowable OCorrupt queries \mathcal{A} issues, as defined in Section 5. 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 5, and treat **Case A**.

In **Case A Game 1**, we define an abort event that triggers if a hash collision occurs. We do so by defining an algorithm $\mathcal{B}_{\text{coll}}$ that computes all hash values honestly, and aborts if there exist two evaluations $(in, \text{H}(in)), (\hat{in}, \text{H}(\hat{in}))$ such that $in \neq \hat{in}$, but $\text{H}(in) = \text{H}(\hat{in})$, outputting this pair to a hash collision challenger if found. 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^*, ζ, b') such that $(i^*, s^*) \neq (i, s)$, or if \mathcal{A} initialises π_i^s such that $\pi_i^s, \text{pid} \neq j$. From now, 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**, we introduce an abort event $\text{abort}_{\text{win}}$ that triggers if the challenger sets $\text{win} \leftarrow 1$ when the test session processes the ciphertext (g^b, c_1) . The rest of the game hops in **Case A** now bound the advantage of \mathcal{A} in causing $\text{abort}_{\text{win}}$ to occur.

Case A Game 5 requires careful consideration: Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ζ', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner π_i^s, pid of the session π_i^s . Thus, we define an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that 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 , replaces the computation of $ck, k_0 \leftarrow \text{KDF}(ck, g^{aB}, 2)$ (in the session π_i^s and its partner) with uniformly random values \tilde{ck}, k_0 , and gives $pk_j = g^u$ to the adversary with all other (honestly generated) public keys. However,

$\mathcal{B}_{\text{PRF-ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol XK, the long-term private keys are used in the following ways: In sessions where the party j acts as the initiator, they compute $ck, k_2 \leftarrow \text{KDF}(ck, g^{xu}, 2)$. Similarly, in sessions where the party acts as the responder, they compute $ck, k_0 \leftarrow \text{KDF}(ck, g^{xu}, 2)$. To simulate this computation, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHu oracle provided by the ms-PRF-ODH challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. We note that $\text{au}^x = 2$, and only after processing (g^b, c_1) will π_i^s output $\varsigma = 2$, and so \mathcal{A} cannot issue a $\text{OCorrupt}(j)$ query before π_i^s processes ciphertext g^b, c_1 . Thus we bound the probability of \mathcal{A} distinguishing this change by the security of the ms-PRF-ODH assumption.

In **Case A Game 6** the challenger replaces the concretely computed values $ck, k_1 \leftarrow \text{KDF}(\tilde{ck}, g^{ab}, 2)$ in π_i^s and its honest partner (if one exists), with uniformly random values \tilde{ck}, \tilde{k}_1 . As by **Game 5**, the input \tilde{ck} is already uniformly random and independent of the protocol execution, distinguishing this game hop can be reduced to the prf security of the KDF. Note that due to this change, the state of π_i^s (containing only \tilde{ck}, \tilde{k}_1 as secrets) is independent of other sessions (making it useless to reveal their states; cf., counters $\text{rp}^i = \text{rp}^x = 2$).

Case A Game 7 proceeds identically to **Game 6**, except that the challenger flips a bit \bar{b} , and uses \bar{b} instead of $\pi_i^s.b_1$ when responding to OEnc or ODec queries from \mathcal{A} directed to sessions π_i^s or π_j^t when using the key \tilde{k}_1 . We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an **aead** challenger, and forwards such OEnc or ODec queries to the **aead** challenger. This change reduces to the **aead** security of the AEAD scheme, and since k_1 is a uniformly random and independent value by **Game 6**, this replacement is sound. The additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{XK_label} \| ad) \| g^B) \| g^a) \| c_0) \| g^b)$. 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 g^b, c_1 . An adversary capable of causing $\text{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 can break the **aead** security of the underlying AEAD scheme, and thus \mathcal{A} has no advantage in causing $\text{abort}_{\text{win}}$ to occur.

$$\begin{aligned} \text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, Case A}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S \cdot \left(\text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} \right. \\ &\quad \left. + \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} \right) \end{aligned}$$

We can now treat **Case B**.

The first four games (**Game 1, 2, 3, 4**) proceed similarly to **Case A**. That is, we abort when a hash-collision is detected, and guess the index (i, s) of the first session π_i^s to set $\text{win} \leftarrow 1$. However, in **Game 3**, we additionally guess the index (j, t) of the intended partner *session*, and abort if our guess is incorrect. **Game 4** still introduces an abort event that occurs if $\text{win} \leftarrow 1$ is set in the test session, and the rest of the game hops bound the advantage of \mathcal{A} in causing the abort event to occur.

Case B Game 5 again requires careful consideration: 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 and by **Game 3**, we know at the beginning of the experiment the index of the honest partner session (j, t) of the session π_i^s . We take a similar approach to **Game 5** of **Case A**. However, in this game we replace the computation of $ck, k_2 \leftarrow \text{KDF}(ck, g^{Ab}, 2)$ with uniformly random and independent values $(\tilde{ck}, \tilde{k}_2)$ in the test session and its honest partner. Specifically, we define an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that initialises a **sym-ms-PRF-ODH** challenger, embeds the DH challenge keyshares 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 , replaces the computation of $ck, k_2 \leftarrow \text{KDF}(ck, g^{Ab}, 2)$ (in the session π_i^s and its partner) with uniformly random values \tilde{ck}, \tilde{k}_2 , and gives $pk_i = g^u$ to the adversary with all other (honestly generated) public keys. However, $\mathcal{B}_{\text{PRF-ODH}}$ 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 in the following ways: In sessions where the party i acts as the initiator, they compute $ck, k_2 \leftarrow \text{KDF}(ck, g^{xu}, 2)$. Similarly, in sessions where the party acts as the responder, they compute $ck, k_0 \leftarrow \text{KDF}(ck, g^{xu}, 2)$. To simulate this computation, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the **ODHu** oracle provided by the **ms-PRF-ODH** challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. However, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $ck, k_0 := \text{KDF}(ck, g^{av})$, where g^a may have been contributed by \mathcal{A} . In this case, in order to compute ck, k_0 , $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the **ODHv** oracle provided by the **sym-ms-PRF-ODH** challenger, specifically querying $\text{ODHv}(ck, g^a)$, which will output $\text{KDF}(ck, g^{av})$. We note that $\text{au}^i = 3$, and only after processing (c_2, c_3) will π_i^s output $\varsigma = 3$, and so \mathcal{A} cannot issue a **OCorrupt**(i) query before π_i^s processes ciphertext c_2, c_3 . Thus we bound the probability of \mathcal{A} distinguishing this change by the security of the **sym-ms-PRF-ODH** assumption. Note that other session states are (and were) independent of π_i^s 's state as g^A is not stored in a state, a collision with g^b would break the above game hop, and \tilde{ck}, \tilde{k}_2 were randomly sampled (cf., counters rp^i, rp^r). **Case B Game 6** proceeds identically to **Game 5**, except that the challenger responds to **OEnc** or **ODec** queries directed to π_i^s or π_j^t outputting $\varsigma = 3$ from \mathcal{A} (i.e. when using the key \tilde{k}_2) and aborts if π_i^s decrypts c_2, c_3 successfully, but it was not output by an honest partner. This changes reduces to the AEAD security of the AEAD scheme. The additional-data field of c_3 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{XK_label} \| ad) \| g^B) \| g^a) \| c_0) \| g^b) \| c_1) \| c_2)$. 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_2, c_3 . Now, \mathcal{A} has no advantage in triggering the event abort_{win} due to π_i^s processing c_2, c_3 .

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

We can now treat **Case C**.

We follow now-standard procedure and define an abort event to trigger when we find 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 $\text{OCorrupt}(j)$ query during the experiment. This allows us to prove the security of all stages ciphertexts.
2. **Case C.2:** \mathcal{A} has issued a $\text{OCorrupt}(j)$ query *after* π_i^s decrypts g^b, c_1 successfully (outputting $\varsigma = 2$). Note that if \mathcal{A} issues a $\text{OCorrupt}(j)$, then $\pi_i^s.\text{fr}_1 \leftarrow 0$, 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 $\varsigma \geq 2$. Note that if \mathcal{A} did not ever issue a $\text{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.1** since $\pi_i^s.\text{fr}_1 = 1$, and we need to capture the security of the additional stages' ciphertext.

In **Case C.1 Game 4**, we replace ck, k_0 by uniformly random \tilde{ck}, \tilde{k}_0 in π_i^s and its honest partner which is reduced to the ms-PRF-ODH assumption (the challenger here acts as in **Case A, Game 5**). Here the session state is again independent of other non-partnered sessions' states. In **Game 5** and **Game 6**, we replace the values $ck, k_1 \leftarrow \text{KDF}(\tilde{ck}, g^{ab}, 2)$ with uniformly random values \tilde{ck}, \tilde{k}_1 , and subsequently replace $ck, k_2 \leftarrow \text{KDF}(\tilde{ck}, g^{Ab}, 2)$ with uniformly random values \tilde{ck}, \tilde{k}_2 via the prf assumption on KDF. Similarly, in **Game 7** we replace $k_{\mathbf{i}}, k_{\mathbf{r}} \leftarrow \text{KDF}(\tilde{ck}, \epsilon, 2)$ with uniformly random values $\tilde{k}_{\mathbf{i}}, \tilde{k}_{\mathbf{r}}$.

In **Case C.1 Game 8** the challenger flips a bit \bar{b} and uses \bar{b} instead of $\pi_i^s.b_1$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0 replaced in **Game 4**). Specifically, the challenger constructs an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\varsigma = 1$ and instead forwards the queries to the AEAD challenger's oracles. Since \tilde{k}_0 is a uniformly random and independent value (by **Game 4**), this change is sound.

Case C.1 Game 9 and **Game 10** proceed identically to **Game 8** but flip and use independent challenge bits when answering queries to OEnc if key k_1 is used in stage $\varsigma = 2$ (**Game 9**) and when key k_2 is used in stage $\varsigma = 3$ (**Game 10**). These changes in \mathcal{A} 's advantage are bound by the advantage in breaking the underlying aead assumption. Finally, in **Game 11** keys $\tilde{k}_{\mathbf{i}}, \tilde{k}_{\mathbf{r}}$ (replaced in **Game 7**) are used in stage $\varsigma = 4$. These changes in \mathcal{A} 's advantage are

bound by the advantage in breaking the underlying `aead` assumption. In **Case C.1, Game 11**, the behaviour of π_i^s is independent on the test bits $\pi_i^s.b_\zeta$ (where $\zeta \geq 1$) and thus \mathcal{A} has no advantage in guessing these challenge bits nor in causing π_i^s to set `win` $\leftarrow 1$.

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

Case C.2 Game 4 requires additional care: Note that by **Game 2**, we know at the beginning of the experiment the index of session π_i^s such that (i, s, ζ', b') is output by the adversary and by **Game 3**, we know at the beginning of the experiment the index of the honest partner session (j, t) of the session π_i^s . We take a similar approach to **Game 5** of **Case A**. However, in this game we replace the computation of $ck, k_1 \leftarrow \text{KDF}(ck, g^{ab}, 2)$ with uniformly random and independent values $(\tilde{ck}, \tilde{k}_1)$ in the test session and its honest partner. Specifically, we define an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that initialises a `sn-PRF-ODH` challenger, embeds the DH challenge keyshares g^u into the ephemeral public-key of party i (g^a), embeds the DH challenge keyshare g^v into the ephemeral public-key of session π_j^t (g^b), and replaces the computation of $ck, k_1 \leftarrow \text{KDF}(ck, g^{ab}, 2)$ (in the session π_i^s and its partner) with uniformly random values \tilde{ck}, \tilde{k}_1 . Note that $\mathcal{B}_{\text{PRF-ODH}}$ can use its internal knowledge of the long-term private keys of party i and party j to compute $(ck, k_0) \leftarrow \text{KDF}(ck, g^{uB}, 2)$ and $(ck, k_2) \leftarrow \text{KDF}(ck, g^{Av}, 2)$. However, $\mathcal{B}_{\text{PRF-ODH}}$ must account for \mathcal{A} to issue `OCorrupt(j)` *after* π_j^t has computed the ciphertext (g^b, c_1) and instead delivering $(g^{b'}, c'_1)$ to π_i^s . To simulate this computation, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the `ODHu` oracle provided by the `sn-PRF-ODH` challenger, specifically querying `ODHu(ck, X)`, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. Thus we bound the probability of \mathcal{A} distinguishing this change by the security of the `sn-PRF-ODH` assumption. Due to this game hop π_i^s 's session state is independent of other non-partnered sessions' states.

In **Case C.2 Game 5** and **Game 6**, we replace $ck, k_2 \leftarrow \text{KDF}(\tilde{ck}, g^{Ab}, 2)$ with uniformly random values \tilde{ck}, \tilde{k}_2 , and subsequently replace $k_i, k_r \leftarrow \text{KDF}(\tilde{ck}, \epsilon, 2)$ with uniformly random values \tilde{k}_i, \tilde{k}_r . **Case C.2, Game 7** proceeds similarly to **Case C.1, Game 9** by encrypting $m_{\bar{b}}$ for a randomly flipped bit \bar{b} when `Enc` and `Dec` would output $\zeta = 2$ (i.e. when using the key \tilde{k}_1). **Case C.1, Game 8** proceeds similarly to **Case C.1, Game 10** by encrypting $m_{\bar{b}}$ for a randomly flipped bit \bar{b} when `Enc` and `Dec` would output $\zeta = 3$ (i.e. when using the key \tilde{k}_2). Finally, **Game 9** proceeds identically to **Case C.1 Game 11** by encrypting $m_{\bar{b}}$ for another randomly flipped bit \bar{b}' when `Enc` and `Dec` would output $\zeta = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r). In **Case C.2, Game 9**, the behaviour of π_i^s is independent of the test bits $\pi_i^s.b_\zeta$ (where $\zeta \geq 2$) and thus \mathcal{A} has no advantage in guessing these challenge bits nor in causing π_i^s to set `win` $\leftarrow 1$. Thus:

$$\text{Adv}_{\mathcal{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, Case C}} \leq \text{Adv}_{\mathcal{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(\max \left\{ \begin{aligned} & (3 \cdot \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} \\ & + 4 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}}), \\ & (2 \cdot \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + 3 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sn-PRF-ODH}}) \end{aligned} \right\} \right).$$

7 Discussion

In our work, we introduce the 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 fACCE model by 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, the examination of which we leave for future work. Similarly, Noise includes *pattern modifiers* that extend the base Noise patterns, for instance, the use of preshared keys, which we also leave for future work.

The aim of our model is explicitly not to propose the next *super-strong* notion of security (since all security properties 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 intertwined design of the specific analyzed protocol (i.e., an atomic channel establishment), it is surprising that all previous ACCE model definitions were heavily influenced by the concept of composing a channel establishment protocol cleanly from key exchange and channel. Consequently, our results systematize and contribute to the understanding of the generic, composition-independent primitive *authenticated and confidential channel establishment*.

Acknowledgments We thank Trevor Perrin, Sebastian Lauer, Sven Schäge, Bertram Poettering, Marc Fischlin, members of the SKECH workshop 2018, and the reviewers for insightful comments and discussions.

Bibliography

- [1] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN: breaking TLS using sslv2. In *USENIX Security*, 2016.
- [2] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO*, 1993.
- [3] Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila. Multi-ciphersuite security of the secure shell (SSH) protocol. In *CCS*, 2014.

- [4] Olivier Blazy, Angèle Bossuat, Xavier Bultel, Pierre-Alain Fouque, Cristina Onete, and Elena Pagnin. Said: Reshaping signal into an identity-based asynchronous messaging protocol with authenticated ratcheting. In *IEEE EuroS&P*, 2019.
- [5] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: relations, instantiations, and impossibility results. In *CRYPTO*, 2017.
- [6] Chris Brzuska and Håkon Jacobsen. A modular security analysis of EAP and IEEE 802.11. In *PKC*, 2017.
- [7] Chris Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of bellare-rogaway key exchange protocols. In *CCS*, 2011.
- [8] Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. An analysis of the EMV channel establishment protocol. In *CCS*, 2013.
- [9] Christina Brzuska, Håkon Jacobsen, and Douglas Stebila. Safely exporting keys from secure channels - on the security of EAP-TLS and TLS key exporters. In *EUROCRYPT*, 2016.
- [10] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, 2003.
- [11] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *USENIX Security*, 2014.
- [12] Shan Chen, Samuel Jero, Matthew Jagielski, Alexandra Boldyreva, and Cristina Nita-Rotaru. 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>.
- [13] Katriel Cohn-Gordon, Cas J. F. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *IEEE EuroS&P*, 2017.
- [14] Jason A. Donenfeld. Wireguard: Next generation kernel network tunnel. In *NDSS*, 2017.
- [15] Benjamin Dowling and Kenneth G. Paterson. A cryptographic analysis of the wireguard protocol. In *ACNS*, 2017.
- [16] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *CCS*, 2015.
- [17] Benjamin Dowling, Paul Rösler, and Jörg Schwenk. Flexible authenticated and confidential channel establishment (facce): Analyzing the noise protocol framework. In *PKC*, 2020.
- [18] Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of google’s QUIC protocol. In *CCS*, 2014.
- [19] Marc Fischlin and Felix Günther. Replay attacks on zero round-trip time: The case of the TLS 1.3 handshake candidates. In *IEEE EuroS&P*, 2017.
- [20] Florian Giesen, Florian Kohlar, and Douglas Stebila. On the security of TLS renegotiation. In *CCS*, 2013.
- [21] Matthew Green. The strange story of “extended random”, 2017. URL <https://blog.cryptographyengineering.com/2017/12/19/the-strange-story-of-extended-random/>.
- [22] Felix Günther and Sogol Mazaheri. A formal treatment of multi-key channels. In *CRYPTO*, 2017.
- [23] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-rtt key exchange with full forward secrecy. In *EUROCRYPT*, 2017.
- [24] Britta Hale, Tibor Jager, Sebastian Lauer, and Jörg Schwenk. Simple security definitions for and constructions of 0-rtt key exchange. In *ACNS*, 2017.

- [25] WhatsApp Inc. WhatsApp encryption overview, 2016. URL <https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf>. White paper.
- [26] Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In *CRYPTO*, 2018.
- [27] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of tls-dhe in the standard model. In *CRYPTO*, 2012.
- [28] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. Authenticated confidential channel establishment and the security of TLS-DHE. *J. Cryptology*, 30(4), 2017.
- [29] Nadim Kobeissi. Noise explorer, 2018. URL <https://noiseexplorer.com/>.
- [30] Nadim Kobeissi, Georgio Nicolas, and Karthikeyan Bhargavan. Noise explorer: Fully automated modeling and verification for arbitrary noise protocols. In *IEEE EuroS&P*, 2019.
- [31] Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of tls-dh and tls-rsa in the standard model. Cryptology ePrint Archive, 2013. <https://eprint.iacr.org/2013/367>.
- [32] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In *CRYPTO*, 2013.
- [33] Brian A. LaMacchia, Kristin E. Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, 2007.
- [34] Yong Li and Sven Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In *CCS*, 2017.
- [35] Benjamin Lipp, Bruno Blanchet, and Karthikeyan Bhargavan. A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol. In *IEEE EuroS&P*, 2019.
- [36] Robert Lychev, Samuel Jero, Alexandra Boldyreva, and Cristina Nita-Rotaru. How secure and quick is quic? provable security and performance analyses. In *IEEE S&P*, 2015.
- [37] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. A modular security analysis of the TLS handshake protocol. In *ASIACRYPT*, 2008.
- [38] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This poodle bites: Exploiting the ssl 3.0 fallback, 2014. URL <https://www.openssl.org/~bodo/ssl-poodle.pdf>.
- [39] Trevor Perrin. The noise protocol framework, 2017. URL <http://noiseprotocol.org/noise.html>. Revision 33.
- [40] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *CRYPTO*, 2018.
- [41] Phillip Rogaway. Authenticated-encryption with associated-data. In *CCS*, 2002.
- [42] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT*, 2006.
- [43] Paul Rösler, Christian Mainka, and Jörg Schwenk. 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 6 (Collision-resistant hash function). A collision-resistant hash function is a deterministic algorithm $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ which, given a bit string m outputs a hash value $w \leftarrow H(m)$ 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 $\text{Adv}_{H, \mathcal{A}}^{\text{coll}} = \Pr [(m, m') \leftarrow \mathcal{A} : H(m) = H(m'), m \neq m']$.

A.2 Pseudo-Random Function Security

Definition 7 (PRF Security). A pseudo-random function family is a collection of deterministic functions $\text{PRF} = \{\text{PRF} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Z}\}$. 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 $k \leftarrow_{\S} \mathcal{K}$ 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 $y_i \leftarrow \text{PRF}(k, m_i)$ (when $b = 0$) or $y_i \leftarrow_{\S} \mathcal{Z}$ (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 $\text{Adv}_{\text{PRF}, \mathcal{A}}^{\text{prf}} = |2 \cdot \Pr[b' = b] - 1|$.

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

A.3 AEAD security

In the following, we will provide a definition of authenticated encryption with associated data as given in [41].

Definition 8 (AEAD scheme). We define an authenticated encryption scheme with associated data (AEAD) as a tuple of algorithms $\text{AEAD} = (\text{kgen}, \text{enc}, \text{dec})$ 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 $c = \text{enc}(k, n, ad, m)$. 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 $m = \text{dec}(k, n, ad, c)$ or \perp . We require that $\text{dec}(k, n, ad, \text{enc}(k, n, ad, m)) = m$ 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 9 (AEAD security). Let $\text{AEAD} = (\text{kgen}, \text{enc}, \text{dec})$ 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

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

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

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

In this section we introduce our definition of a “dual” PRF-ODH assumption. We begin with the generic definition of a PRF-ODH assumption.

Definition 10 (Generic PRF-ODH Assumption). Let G be a cyclic group of order q with generator g . Let $\text{PRF} : G \times \mathcal{M} \rightarrow \mathcal{K}$ 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: $l, r \in \{\mathbf{n}, \mathbf{s}, \mathbf{m}\}$ indicating how often the adversary is allowed to query “left” and “right” oracles (ODHu and ODHv), where \mathbf{n} indicates that no query is allowed, \mathbf{s} that a single query is allowed, and \mathbf{m} that multiple queries are allowed to the respective oracle. Consider the following security game $\mathcal{G}_{\text{PRF}, G, p, \mathcal{A}}^{\text{lr-PRF-ODH}}$ between a challenger \mathcal{C} and an adversary \mathcal{A} .

<p>Init(1^λ)</p> <p>$k \leftarrow_{\S} \text{kgen}$</p> <p>$b \leftarrow_{\S} \{0, 1\}$</p> <p>$S \leftarrow \{\}$</p> <p>$b' \leftarrow_{\S} \mathcal{A}^{\text{Enc}(), \text{Dec}()}$</p> <p>return ($b = b'$)</p> <p>Dec($n, ad, c$)</p> <p>if ($b = 0$):</p> <p> return \perp</p> <p>if ($\text{dec}(k, n, ad, c) \neq \perp$ $\wedge ((n, ad, c) \notin S)$):</p> <p> return m</p> <p>else return \perp</p>	<p>Enc(n, ad, m_0, m_1)</p> <p>$c_0 = \text{enc}(k, n, ad, m_0)$</p> <p>$c_1 = \text{enc}(k, n, ad, m_1)$</p> <p>if ($c_0 = \perp$) \vee ($c_1 = \perp$):</p> <p> return \perp</p> <p>else</p> <p> $S \leftarrow S \cup \{(n, ad, c_b)\}$</p> <p> return c_b</p>
---	---

Fig. 4: Security experiment for AEAD schemes $\text{AEAD} = (\text{kgen}, \text{enc}, \text{dec})$.

1. The challenger \mathcal{C} samples $u \leftarrow_{\S} \mathbb{Z}_p$ and provides g, g^u to the adversary \mathcal{A} .
2. If $l = m$, \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 $y \leftarrow \text{PRF}_\lambda(S^u, x)$ and returns y .
3. Eventually, \mathcal{A} issues a challenge query x^* . The challenger \mathcal{C} samples $v \leftarrow_{\S} \mathbb{Z}_p$ 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 $b \leftarrow_{\S} \{0, 1\}$ uniformly at random, computes $y_0 = \text{PRF}_\lambda(g^{uv}, x^*)$, and samples $y_1 \leftarrow_{\S} \{0, 1\}^\lambda$ 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 $y \leftarrow \text{PRF}_\lambda(S^u, x)$.
 - 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 $y \leftarrow \text{PRF}_\lambda(T^v, x)$.
5. At some point, \mathcal{A} outputs a guess bit $b' \in \{0, 1\}$.

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

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

Intuitively, the lr-PRF-ODH assumption holds if the advantage $\text{Adv}_{\text{PRF}, G, p, \mathcal{A}}^{\text{lr-PRF-ODH}}$ 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 kc^i, kc^r) and resistance against randomness reveal (indicated by counters $\text{eck}, \text{r1}^i, \text{r1}^r$). We describe these counters below:

1. kc^ρ defines the stage required for KCI resistance of ρ to be reached. It is hard, for a stage $\varsigma \geq \text{kc}^\rho$, 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 $\varsigma \geq \text{fs}$, 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. **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 [33])²⁴.
3. **r1 ρ** 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 (**fs**, **eck**, **r1 ρ**) 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.

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 4) 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 $a \leq |\pi_j^t.T_d|$, $b \leq |\pi_i^s.T_d|$ such that $a > 0$ if $\pi_i^s.\rho = \mathbf{i}$ and $b > 0$ if $\pi_i^s.\rho = \mathbf{r}$ then $\forall 0 \leq \alpha < a : (\pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha]) \wedge \forall 0 \leq \beta < b : (\pi_i^s.T_d[\beta] = \pi_j^t.T_e[\beta])$).

We provide a pseudocode definition of function $\text{PrevPartner}(i, s)$ 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 **Onit**, **OEnc**, **ODec**, **OCorrupt**, **OReveal** – the ability to reveal sessions' random coins via the oracle **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 **Onit**, **OEnc**, **OCorrupt**, **OReveal** of the full model towards the reduced model in the main body (see Section 5) is that $\text{Fresh}_{full}()$ is invoked to update the freshness flags instead of $\text{Fresh}_{fs}()$. The computation of the freshness flag via $\text{Fresh}_{full}()$ 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.

- **ODec**(i, s, c) triggers invocation of $\text{Dec}(sk_i, \pi_i^s.st, c) \rightarrow_{[\pi_i^s.rand]} (st', m, \varsigma)$ 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.

fs, eck, r1 ρ : Since decryption can change the honesty of partners, the freshness flags are updated regarding corruptions and the reveal of random coins by invoking $\text{Fresh}_{full}()$ (see Table 3 and Figure 5).

au, 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., $\varsigma < \mathbf{au}^{\pi_i^s.\bar{\rho}}$), then all following stages are marked un-fresh until authentication will be reached ($\pi_i^s.fr_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^* < \mathbf{au}^{\pi_i^s.\bar{\rho}}$), since this is a (temporarily) trivial impersonation of the partner towards π_i^s .

²⁴ 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. [11, 21]) or a random oracle, used to derive random coins on a low entropy input (such as the current time).

2. is reached with this ciphertext (i.e., $\varsigma = \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$), and
 - (a) intended partner $\pi_i^s \cdot \text{pid}$ is corrupted ($\text{corr}_{\pi_i^s \cdot \text{pid}} = 1$), then all following stages are marked un-fresh ($\pi_i^s \cdot \text{fr}_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is a trivial impersonation of the partner towards π_i^s .
 - (b) intended partner $\pi_i^s \cdot \text{pid}$ is not corrupted, party i is corrupted, and KCI resistance is not reached yet (i.e., $\varsigma < \mathbf{kc}^{\pi_i^s \cdot \rho}$, $\text{corr}_{\pi_i^s \cdot \text{pid}} \neq \text{corr}_i = 1$), then all following stages are marked un-fresh until KCI resistance would be reached ($\pi_i^s \cdot \text{fr}_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^* < \mathbf{kc}^{\pi_i^s \cdot \rho}$), since this is a (temporarily) trivial *key compromise impersonation* of the partner towards π_i^s .²⁵
 3. was reached before, but KCI resistance is not reached yet, and party i is corrupted (i.e., $\mathbf{au}^{\pi_i^s \cdot \bar{\rho}} < \varsigma < \mathbf{kc}^{\pi_i^s \cdot \rho}$, $\text{corr}_i = 1$), then all following stages are marked un-fresh until KCI resistance would be reached ($\pi_i^s \cdot \text{fr}_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^* < \mathbf{kc}^{\pi_i^s \cdot \rho}$), since this is a (temporarily) trivial *key compromise impersonation* of the partner towards π_i^s .²⁵
 4. is reached or was reached before, and the session's random coins were revealed (i.e., $\mathbf{au}^{\pi_i^s \cdot \bar{\rho}} \leq \varsigma$, $\pi_i^s \cdot \text{rr} = 1$), then all following stages are marked un-fresh ($\pi_i^s \cdot \text{fr}_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is a trivial impersonation of the partner towards π_i^s .²⁶
- 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 $\pi_i^s \cdot \text{pid}$ 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 ($\varsigma = \mathbf{kc}^{\pi_i^s \cdot \rho}$), and the intended partner is corrupted ($\text{corr}_{\pi_i^s \cdot \text{pid}} = 1$), then all following stages are marked un-fresh ($\pi_i^s \cdot \text{fr}_{\varsigma^*} \leftarrow 0$ for all $\varsigma \leq \varsigma^*$), since this is a trivial impersonation of the partner towards π_i^s .

Rewarding real attacks:

- au, 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 $\text{win} \leftarrow 1$ 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 ($\pi_i^s \cdot \text{fr}_{\varsigma} = 1$), π_i^s 's randomness was not revealed ($\pi_i^s \cdot \text{rr} = 0$)²⁶, and
1. this is the first authenticated ciphertext ($\varsigma = \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$), the intended partner $\pi_i^s \cdot \text{pid}$ is not yet corrupted, and, if KCI resistance was not reached yet, then party i is not yet corrupted ($\text{corr}_{\pi_i^s \cdot \text{pid}} = 0$ and if $\varsigma < \mathbf{kc}^{\pi_i^s \cdot \rho}$ then $\text{corr}_i = 0$), or
 2. this is a later authenticated ciphertext ($\varsigma > \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$) and, if KCI resistance was not reached yet, then party i is not yet corrupted (if $\varsigma < \mathbf{kc}^{\pi_i^s \cdot \rho}$ then $\text{corr}_i = 0$).
- **ORevealRandomness**(i, s) \rightarrow *rand* outputs the randomness $\pi_i^s \cdot \text{rand}$ sampled by party i in its session π_i^s . The session is marked $\pi_i^s \cdot \text{rr} \leftarrow 1$ and the freshness flags are updated by invoking $\text{Fresh}_{full}()$.

B.2 Freshness regarding Corruption and Reveal of Randomness

Based on the definition of the counters \mathbf{fs} , \mathbf{eck} , $\mathbf{r1}^i$, and $\mathbf{r1}^F$, the effects of corruptions and the revelation of sessions' random coins can be mapped onto the freshness flag $\pi_i^s \cdot \text{fr}_{\varsigma}$ 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., $\text{corr}_{\pi_i^s \cdot \text{pid}}$), the revelation of the sessions' own random coins (i.e., $\pi_i^s \cdot \text{rr}$), and the revelation of honest partners' random coins (i.e., $\pi_j^t \cdot \text{rr}$; $j = \pi_i^s \cdot \text{pid}$ by definition) need to be considered. Not only do the current honest partners affect the 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 \mathbf{fs} , \mathbf{eck} , $\mathbf{r1}^{\rho}$. The oracle description above updates the freshness flags based on this table via function $\text{Fresh}_{full}()$ (defined in Figure 5). Recall that the two last cases in Figure 5 consider replayed ciphertexts that introduce insecure sessions.

²⁵ Obviously when KCI resistance is reached, key compromise impersonation is no longer considered trivial.

²⁶ We discuss on this trivial attack and its necessity, in Appendix C.6. One can think of it as a KCI attack based on the receiver's ephemeral secrets (instead of its long-term secrets).

	$\zeta < \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$	$\zeta = \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$ $\zeta < \mathbf{kc}^{\pi_i^s \cdot \rho}$	$\zeta = \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$ $\zeta = \mathbf{kc}^{\pi_i^s \cdot \rho}$	$\zeta > \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$ $\zeta < \mathbf{kc}^{\pi_i^s \cdot \rho}$	$\zeta > \mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$ $\zeta = \mathbf{kc}^{\pi_i^s \cdot \rho}$	$\zeta > \mathbf{kc}^{\pi_i^s \cdot \rho}$
$\text{corr}_i = \text{corr}_{pid} = 0$	$\mathbf{au}^{\pi_i^s \cdot \bar{\rho}}$	win	win	win	win	win
$\text{corr}_i = 0 \wedge \text{corr}_{pid} = 1$	∞	∞	∞	win	win	win
$\text{corr}_i = 1 \wedge \text{corr}_{pid} = 0$	$\mathbf{kc}^{\pi_i^s \cdot \rho}$	$\mathbf{kc}^{\pi_i^s \cdot \rho}$	win	$\mathbf{kc}^{\pi_i^s \cdot \rho}$	win	win
$\text{corr}_i = \text{corr}_{pid} = 1$	∞	∞	∞	∞	∞	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 \mathbf{ctr} denotes the first stage for which the freshness flag is not erased: $\forall \zeta \leq \zeta^* < \mathbf{ctr} : \pi_i^s \cdot \text{fr}_{\zeta^*} \leftarrow 0$. A successful attack is denoted by cells containing win, meaning that if $\pi_i^s \cdot \text{fr}_{\zeta} = 1$ and the state st output by π_i^s 's invocation of Dec is not \perp , then $\text{win} \leftarrow 1$ is set. Note that always $\mathbf{au}^{\bar{\rho}} \leq \mathbf{kc}^{\rho}$.

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:

Freshness of π_i^s based on:		$\text{corr}_{\pi_i^s \cdot pid} = 1$	$\pi_j^t \cdot rr = 1$	$\pi_j^t \cdot rr = 1$ $\wedge \text{corr}_{\pi_i^s \cdot pid} = 1$
		$\min(\mathbf{rl}^{\pi_i^s \cdot \bar{\rho}}, \mathbf{fs})$	$\min(\mathbf{rl}^{\pi_i^s \cdot \rho}, \mathbf{eck})$	∞
$\text{corr}_i = 1$	$\min(\mathbf{rl}^{\pi_i^s \cdot \rho}, \mathbf{fs})$	\mathbf{fs}	$\mathbf{rl}^{\pi_i^s \cdot \rho}$	∞
$\pi_i^s \cdot rr = 1$	$\min(\mathbf{rl}^{\pi_i^s \cdot \bar{\rho}}, \mathbf{eck})$	$\mathbf{rl}^{\pi_i^s \cdot \bar{\rho}}$	\mathbf{eck}	∞
$\pi_i^s \cdot rr = 1$ $\wedge \text{corr}_i = 1$	∞	∞	∞	∞

Table 3: Freshness of session π_i^s is defined over corruptions of both parties $i, \pi_i^s \cdot pid$ 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.

In Appendix C.7 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 5 except that also counters ($\mathbf{kc}^i, \mathbf{kc}^r, \mathbf{eck}, \mathbf{rl}^i, \mathbf{rl}^r$) are considered:

Definition 12 (Advantage in Breaking Flexible ACCE). *An adversary \mathcal{A} breaks a flexible ACCE protocol fACCE with authentication stages ($\mathbf{au}^i, \mathbf{au}^r$), key compromise impersonation resilience stages ($\mathbf{kc}^i, \mathbf{kc}^r$), forward-secrecy stage \mathbf{fs} , eCK security stage \mathbf{eck} , stages for confidentiality based on long-term secrets and random coins respectively ($\mathbf{rl}^i, \mathbf{rl}^r$), and replayability resistance stages ($\mathbf{rp}^i, \mathbf{rp}^r$), when \mathcal{A} terminates and outputs (i, s, ζ, b') , if there either exists a session π_i^s such that $\pi_i^s \cdot b_{\zeta} = b'$, and $\pi_i^s \cdot \text{fr}_{\zeta} = 1$ (which we subsume as event \mathbf{guess}), or $\text{win} = 1$. We define the advantage of an adversary \mathcal{A} breaking a flexible ACCE protocol fACCE as $\text{Adv}_{\mathcal{A}}^{\text{fACCE}} = (2 \cdot \Pr[\mathbf{guess}] - 1) + \Pr[\text{win} = 1]$.*

Intuitively, a flexible ACCE protocol fACCE is secure if it is correct and $\text{Adv}_{\mathcal{A}}^{\text{fACCE}}$ 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.

```

Freshfull( $\cdot$ ):
For all  $i \in [n_P]$ , for all  $s \in [n_S]$ :
   $\mathbf{ctr} \leftarrow \min(\zeta^* : \pi_i^s.fr_{\zeta^*} = 1)$ 
  If  $corr_{\pi_i^s.pid} = 1$ :
     $\mathbf{ctr} \leftarrow_{\max} \min(\mathbf{rl}^{\pi_i^s.\bar{\rho}}, \mathbf{fs})$ 
  If  $\exists(j, t) \in \text{PrevPartner}(i, s) : \pi_j^t.rr = 1$ :
     $\mathbf{ctr} \leftarrow_{\max} \min(\mathbf{rl}^{\pi_i^s.\rho}, \mathbf{eck})$ 
  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  $\pi_i^s.rr = 1$ ):
     $\mathbf{ctr} \leftarrow \infty$ 
  If  $corr_i = 1$ :
     $\mathbf{ctr} \leftarrow_{\max} \min(\mathbf{rl}^{\pi_i^s.\rho}, \mathbf{fs})$ 
    If  $corr_{\pi_i^s.pid} = 1$ :
       $\mathbf{ctr} \leftarrow_{\max} \mathbf{fs}$ 
    If  $\exists(j, t) \in \text{PrevPartner}(i, s) : \pi_j^t.rr = 1$ :
       $\mathbf{ctr} \leftarrow_{\max} \mathbf{rl}^{\pi_i^s.\rho}$ 
  If  $\pi_i^s.rr = 1$ :
     $\mathbf{ctr} \leftarrow_{\max} \min(\mathbf{rl}^{\pi_i^s.\bar{\rho}}, \mathbf{eck})$ 
    If  $corr_{\pi_i^s.pid} = 1$ :
       $\mathbf{ctr} \leftarrow_{\max} \mathbf{rl}^{\pi_i^s.\bar{\rho}}$ 
    If  $\exists(j, t) \in \text{PrevPartner}(i, s) : \pi_j^t.rr = 1$ :
       $\mathbf{ctr} \leftarrow_{\max} \mathbf{eck}$ 
  If  $\pi_i^s.T_e[0] \in Rpl \wedge \pi_i^s.\rho = i$ :
     $\mathbf{ctr} \leftarrow \infty$ 
  If  $\pi_i^s.T_d[0] \in Rpl \wedge \pi_i^s.\rho = r$ :
     $\mathbf{ctr} \leftarrow \infty$ 
   $\pi_i^s.fr_{\zeta^*} \leftarrow 0$  for all  $\zeta^* < \mathbf{ctr}$ 

```

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 $\text{PrevPartner}(i, s)$ see Figure 7.

C.1 Restrictions of Generality

As described in the main body of our paper, we introduced slight simplifications and made definitional choice in favor of Noise in case these choices do not restrict our consideration of a generic concept of an *Authenticated Confidential Channel Establishment* protocol. For clarity and transparency, we describe and explain these choices below.

Our syntax only considers associated data for session initialization (but not for each transmitted payload). As described before, associated data for each payload can trivially be added to the syntax and can generically be provided by a construction (without requiring further hardness assumptions). Furthermore, our syntax treats the security level as an output stage counter. In practice, an input that defines security for payload, which is to be transmitted, might be more useful than an output (as the transmission is triggered already when our stage counter output tells the application under which security this is conducted). For proving the security, it is nevertheless sufficient to consider the security level as an output of Enc and Dec.

We define partnering in a matching conversations-style that requires uniqueness of partnering as soon as possible. As a result, we only consider channel establishment protocols that enforce a FIFO-channel and prevent bit flips in the session transcript and (unpreventable) replay attacks. As noted before, it is important for a generic model to utilize natural definitions (which matching conversations in contrast to ambiguous notions like *session IDs* are). Nevertheless, we discuss alternative partnering approaches in Appendix C.2.

Our security game assumes a public key infrastructure (which is common in ACCE and AKE notions but not necessary; it only simplifies definitions). We assume that security properties are monotonically extended by the protocols (which again dramatically simplifies our definitions). These two latter restrictions as well as our limited choice of security properties depict an exemplary setting but can be extended easily. As explained before, the restrictions in the considered security definitions comprise that we neither consider forward-secrecy with respect to the individual session participants' long-term secrets, nor

inner-session forward-secrecy, we disregard trust-on-first-use authentication, we only regard the leakage of randomness rather than the manipulation of randomness, and we base our definition of the impact of replay attacks on the dependence of session secrets to the first transmitted ciphertext in a session (independent of the ciphertext’s origin, destination, reached authentication, . . .). We discuss many of these choices below and emphasize that all these restrictions were introduced as choices among equivalent alternatives in order to provide a useful, comprehensive, and natural model.

The only real dependency on Noise lays within our full model’s treatment of authentication under randomness reveal: We say that authentication of the counter party is not required to be reached in case the own random coins are leaked. This (preventable) KCI-related trivial attack is introduced because Noise’s authentication (based on static-with-ephemeral Diffie-Hellman key exchanges) does not prevent it, which we discuss in Appendix C.6.

C.2 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 [34]). 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 [10] relaxes CCA security²⁷. 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.3 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, . . . 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 $\text{RT}(\text{au}^r) = 1$ and $\text{RT}(\text{au}^i) = 1.5$ (with $\text{au}^r = 2, \text{au}^i = 3$). 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 $\text{RT}(\text{fs}) = 1, \text{fs} = 2$.

²⁷ This idea stems from discussions with Bertram Poettering at SKECH Workshop 2018.

C.4 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:

In order to analyze protocols that allow the deterministic update of an established channel key (as in TLS 1.3), Günther and Mazaheri [22] 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 [26] and Poettering and Rösler [40] propose strong models for defining security in the presence of adversaries who can reveal the local state. Their model requires 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 be considered in channel establishment security notions).

C.5 On the Necessity of One Holistic Model

As noted before, we here explain why this splitting the security definition regarding the considered security properties 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 (e.g., trivial attacks against authenticity invalidate confidentiality challenges), 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 Figure 6 that (without a proof) provides authentication under secure randomness (i.e., $\text{au}^1 = x < \infty$, $x \leq \text{eck} = y \leq \infty$) and confidentiality under randomness reveal (i.e., $\text{eck} = z < x < \infty$, $\text{au}^1 = \infty$), but not both properties simultaneously (i.e., there is an attack for defining $\text{au}^1 = x$ and $\text{eck} = z$ 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.

The depicted protocol deterministically encrypts payload with the first two ciphertexts and only afterwards authenticates the transcript with a signature (from an ephemeral signing key pair that is authenticated by the AEAD ciphertexts before). Under the assumption that the signature scheme reveals the ephemeral signing 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.

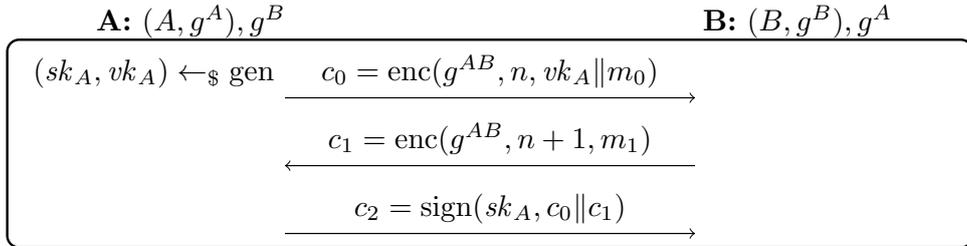


Fig. 6: Example protocol that provides client authentication after receiving ciphertext c_2 if randomness is revealed only afterwards (i.e., $\mathbf{au}^i = 3, \mathbf{eck} > 3$) but that provides confidentiality and no authentication under revealed randomness from the first ciphertext on already (i.e., $\mathbf{eck} = 1, \mathbf{au}^i = \infty$). 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.6 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 sessions take a long time. For our setting, we regard the precision of our authentication consideration sufficient.²⁸

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.

²⁸ 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.

C.7 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., [11, 21]). 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 [33], 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 three 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 `rl ρ`).

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.

D Correctness and Partnering Definitions in Pseudocode

For full precision and more clarity, we provide pseudocode descriptions of definitions 3 (correctness), 4 (honest partner), and 11 (previous honest partner) in Figure 7. An fACCE scheme is correct if $\Pr[\text{Correct}(b_i, b_r) \rightarrow 1] = 0$ for some $(b_i, b_r) \in \{0, 1\}^2$ (depending on for whom the scheme provides authentication).

The correctness game allows an unbounded adversary to interact 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) honest partners*. As described in Definitions 4 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.

E Full Proofs

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

<p>Game Correct(b_i, b_r)</p> <p>$gen \leftarrow 0; init \leftarrow 0$ $sk_i \leftarrow \perp; pk_i \leftarrow \perp; sk_r \leftarrow \perp; pk_r \leftarrow \perp$ $st_i \leftarrow \perp; st_r \leftarrow \perp; \varsigma_i^{\max} \leftarrow 0; \varsigma_r^{\max} \leftarrow 0$ $MS_i \leftarrow \perp; MS_r \leftarrow \perp; T_i \leftarrow \perp; T_r \leftarrow \perp$ If $b_i = 1$: $(sk_i, pk_i) \leftarrow_{\S} \text{KGen}$ If $b_r = 1$: $(sk_r, pk_r) \leftarrow_{\S} \text{KGen}$ Invoke $\mathcal{A}(sk_i, pk_i, sk_r, pk_r)$ Abort with 0</p> <p>Oracle Init(ad)</p> <p>If $gen = 0 \vee init = 1 \vee ad \notin \{0, 1\}^*$: Abort with 0 $init \leftarrow 1$ $st_i \leftarrow_{\S} \text{Init}(sk_i, pk_i, i, ad)$ $st_r \leftarrow_{\S} \text{Init}(sk_r, pk_r, r, ad)$ Return st_i, st_r</p> <p>Oracle Enc(ρ, m)</p> <p>If $init = 0 \vee \rho \notin \{i, r\} \vee m \notin \{0, 1\}^*$: Abort with 0 $(st_\rho, c, \varsigma) \leftarrow_{\S} \text{Enc}(sk_\rho, st_\rho, m)$ If $\varsigma < \varsigma_\rho^{\max}$: Abort with 0 $\varsigma_\rho^{\max} \leftarrow \varsigma$ $MS_\rho \leftarrow MS_\rho \parallel (m, \varsigma)$ $T_\rho \leftarrow T_\rho \parallel c$ Return (st_ρ, c, ς)</p> <p>Oracle Dec(ρ)</p> <p>If $init = 0 \vee \rho \notin \{i, r\}$: Abort with 0 $c \parallel T_\rho \leftarrow T_\rho$ If $c \notin \{0, 1\}^* \setminus \{\epsilon\}$: Abort with 0 $(st_\rho, m, \varsigma) \leftarrow_{\S} \text{Dec}(sk_\rho, st_\rho, c)$ $(m^*, \varsigma^*) \parallel MS_\rho \leftarrow MS_\rho$ If $m \neq \perp \wedge (m, \varsigma) \neq (m^*, \varsigma^*)$: Abort with 1</p>	<p>Partner(i, s)</p> <p>$partners \leftarrow \emptyset$ 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 $\pi_i^s.pid = j \wedge \pi_j^t.pid = i$ $\wedge \pi_i^s.\rho \neq \pi_j^t.\rho$ $\wedge \pi_i^s.ad = \pi_j^t.ad$ $\wedge \forall 0 \leq \alpha < a : \pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha]$ $\wedge \forall 0 \leq \beta < b : \pi_j^t.T_e[\beta] = \pi_i^s.T_d[\beta]$ $\wedge (\pi_i^s.\rho = i \wedge a > 0 \vee \pi_i^s.\rho = r \wedge b > 0)$: $partners \leftarrow partners \cup \{(j, t)\}$ If $\pi_i^s.T_d \neq \perp \wedge partners > 1$: Return \emptyset Return $partners$</p> <hr/> <p>PrevPartner(i, s)</p> <p>$partners \leftarrow \emptyset$ 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 $\pi_i^s.pid = j \wedge \pi_j^t.pid = i$ $\wedge \pi_i^s.\rho \neq \pi_j^t.\rho$ $\wedge \pi_i^s.ad = \pi_j^t.ad$ $\wedge \exists 0 \leq a^* \leq a : \forall 0 \leq \alpha < a^* : \pi_i^s.T_e[\alpha] = \pi_j^t.T_d[\alpha]$ $\wedge \exists 0 \leq b^* \leq b : \forall 0 \leq \beta < b^* : \pi_j^t.T_e[\beta] = \pi_i^s.T_d[\beta]$ $\wedge (\pi_i^s.\rho = i \wedge a^* > 0 \vee \pi_i^s.\rho = r \wedge b^* > 0)$: $partners \leftarrow partners \cup \{(j, t)\}$ Return $partners$</p>
---	--

Fig. 7: Pseudocode definitions of correctness, honest partnering, and previous honest partnering (see definitions 3, 4, and 11). It is assumed that the ‘ \parallel ’ 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.1 N pattern

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

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

Proof.

First, recall that the adversary breaks authentication (and thus, sets $\text{win} \leftarrow 1$) 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

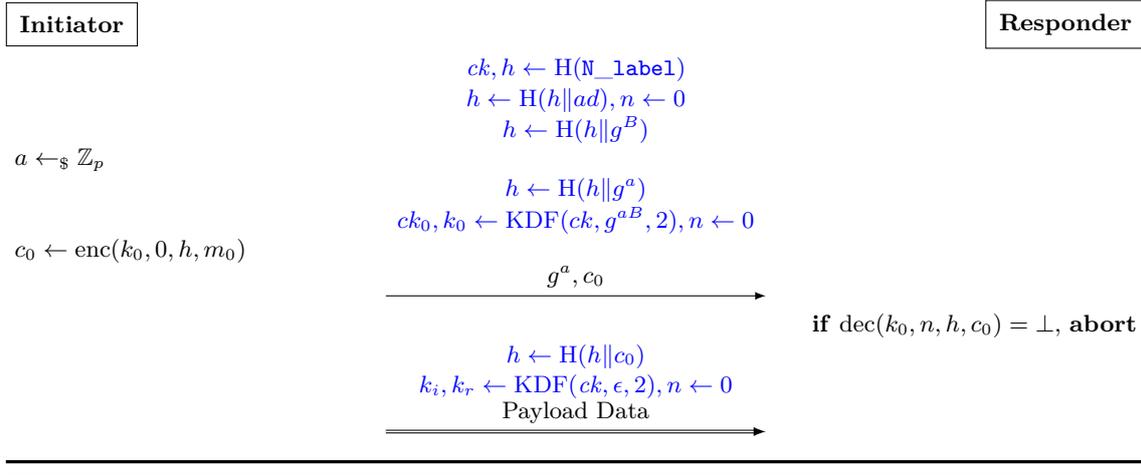


Fig. 8: Noise Pattern $\mathbb{N} : \leftarrow \mathbf{s}, \dots, \rightarrow \mathbf{e}, \mathbf{es}$.

the security experiment, it is required that if $\text{win} \leftarrow 1$ in the responder session in any given stage, the stage must still be fresh. Since the initiator never authenticates (i.e., $\text{au}^1 = \infty$), the adversary injecting the ciphertext from a non-honest partner would cause $\pi_i^s \cdot \text{fr}_\zeta \leftarrow 0 \forall \zeta$, and thus $\Pr[\text{win} \leftarrow 1] = 0$. 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 \mathbb{N} , and thus the behavior of a responder session is independent of its challenge bits). Note that in the Noise Protocol \mathbb{N} where \mathcal{A} terminates and outputs a tuple (i, s, ζ', b') , if $\pi_i^s \cdot \rho = i$, and $\pi_i^s \cdot \text{fr}_1 = 1$ then \mathcal{A} cannot issue either a $\text{OCorrupt}(\pi_i^s \cdot \text{pid})$, nor a $\text{ORevealRandomness}(i, s)$ query, as $\mathbf{fs} = \mathbf{eck} = \mathbf{r1}^r = \infty$, and $\text{au}^r = \infty$. 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:

$$\text{Adv}_{\mathbb{N}, n_P, n_S, \mathcal{A}}^{\text{fACCE}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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 $\pi_i^s \cdot \text{pid} \neq j$. Thus:

$$\text{Adv}(\text{break}_2) = n_P (\text{Adv}(\text{break}_3)).$$

Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 in the test session π_i^s and its honest partner (if one exists) by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ 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, ζ', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner $\pi_i^s \cdot \text{pid}$ of the session π_i^s . Thus, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\text{OCorrupt}(j)$ query, as $\mathbf{eck} = \mathbf{fs} = \mathbf{r1}^r = \infty$ and $\mathbf{r1}^1 = 1$. However, $\mathcal{B}_{\text{PRF-ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol \mathbb{N} , the long-term private keys are used to compute the following:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary.

In the second case, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHu oracle provided by the ms-PRF-ODH challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. We note that $\mathbf{fs}, \mathbf{r1}^x = \infty$, and as such $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{OCorrupt}(j)$ query nor answer a $\text{ORevealRandomness}(i, s)$ query.

Since $\mathbf{rp} = \infty$, however, $\mathcal{B}_{\text{PRF-ODH}}$ also has to simulate sessions $\pi_j^{t^*}$ where g^v, c_0 (the ciphertext sent by π_i^s) is replayed to $\pi_j^{t^*}$. $\mathcal{B}_{\text{PRF-ODH}}$ simply sets the computation of ck, k_0 in such sessions to \tilde{ck}, \tilde{k}_0 to ensure consistency. Thus we have:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0 replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an aead challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when $\varsigma = 1$, (i.e. when using the key \tilde{k}_0) and π_j^t is the honest partner of π_i^s . $\mathcal{B}_{\text{aead}}$ instead queries $\text{Enc}(n, h, m_0, m_1)$ and $\text{Dec}(n, h, c)$ to the aead challenger's oracles.

It is useful to note here that the hash h used as the ad field here contains all ciphertexts sent between the sessions to this point. Since we forbid hash collisions in **Game 1**, if the initiator and responder have different views of the current transcript, then they will compute different hash values h and h' , and thus will abort when decrypting a ciphertext containing the wrong ad field.

Since \tilde{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,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_i replaced in **Game 5**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an aead challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when $\varsigma = 2$, (i.e. when using the key \tilde{k}_i) and π_j^t is the honest partner of π_i^s . $\mathcal{B}_{\text{aead}}$ instead queries $\text{Enc}(n, ad, m_0, m_1)$ and $\text{Dec}(n, ad, c)$ to the aead challenger's oracles.

Since \tilde{k}_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,

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

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:

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

E.2 NK pattern

Theorem 3. *The Noise protocol NK is flexible-ACCE-secure protocol with authentication levels $\mathbf{au} = (\infty, 2)$, forward-secrecy $\mathbf{fs} = 2$, KCI resistance $\mathbf{kc} = (\infty, 2)$, one-way-randomness-security $\mathbf{rl} = (1, \infty)$, eCK security $\mathbf{eck} = \infty$, and replay resistance $\mathbf{rp} = (2, 2)$. That is, for an adversary \mathcal{A} against the flexible ACCE security game (defined in section 5) one can define efficient adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of H , $\mathcal{B}_{\text{PRF-ODH}}$ against the PRF-ODH assumptions ms-PRF-ODH and nn-PRF-ODH with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the PRF security of KDF with:*

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

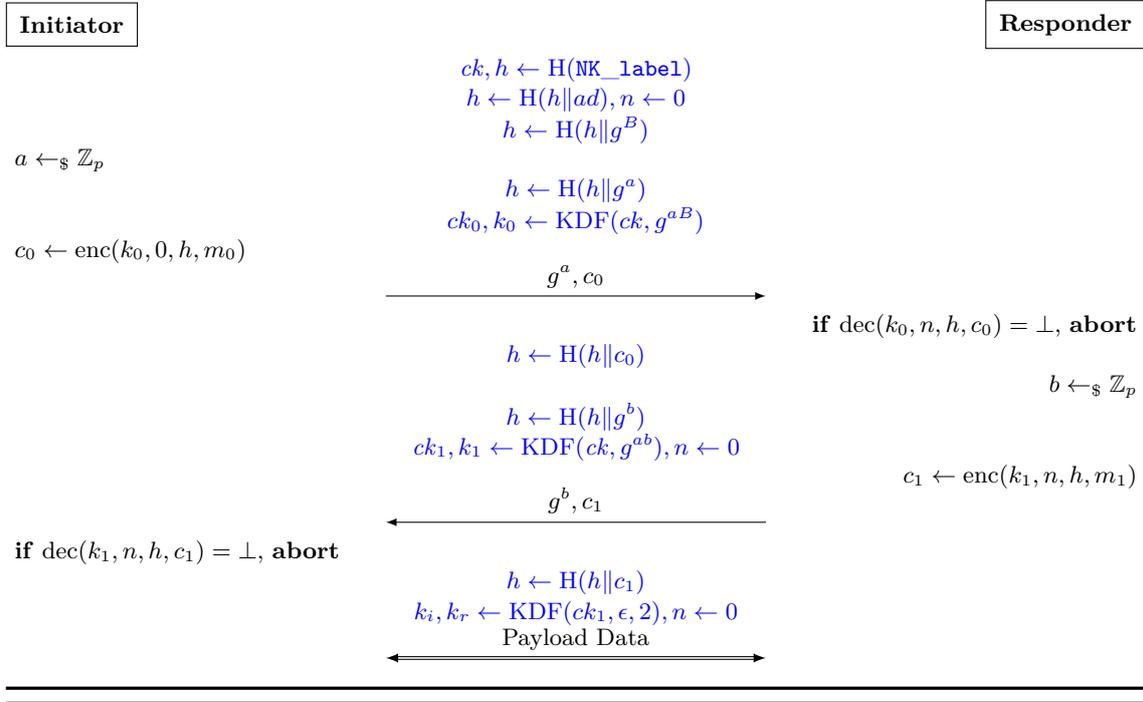


Fig. 9: Noise Pattern NK $\leftarrow s, \dots, \rightarrow e, es$.

Proof.

We split the analysis into the following three cases:

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

It is clear that

$$\text{Adv}_{\text{NK}, n_P, n_S, \mathcal{A}}^{\text{fACCE}} \leq \text{Adv}_{\text{NK}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CA}} + \text{Adv}_{\text{N}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CB}} + \text{Adv}_{\text{NK}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CC}}$$

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

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

$$\text{Adv}_{\text{NK}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CA}} = \text{Adv}(\text{break}_0).$$

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

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

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:

$$\text{Adv}(\text{break}_1) = n_P n_S (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P (\text{Adv}(\text{break}_3)).$$

Case A, Game 4: In this game, we define an abort event $\text{abort}_{\text{win}}$ if the session π_i^s sets the status $\text{win} \leftarrow 1$ 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 $\pi_i^s.\rho = \mathbf{i}$ and $\pi_i^s.b_2 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathbf{i}$ 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 $\text{win} = 1$. 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 $\text{win} \leftarrow 1$ 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 $\text{Adv}(\text{abort}_{\text{win}})$. Thus:

$$\text{Adv}(\text{break}_3) \leq \text{Adv}(\text{abort}_{\text{win}}).$$

Case A, Game 5: In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a ms-PRF-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, $\mathcal{B}_{\text{PRF-ODH}}$ 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. However, $\mathcal{B}_{\text{PRF-ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol NK, the long-term private keys are used to compute the following:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the **ODHu** oracle provided by the **ms-PRF-ODH** challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. 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** $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{OCorrupt}(j)$ query. In addition, since $\text{r1}^i = \infty$, $\text{eck} = \infty$, $\mathcal{B}_{\text{PRF-ODH}}$ also never has to answer a $\text{ORevealRandomness}(i, s)$ query. However $\text{rp}^i, \text{rp}^x = 1$, and $\mathcal{B}_{\text{PRF-ODH}}$ also has to simulate sessions $\pi_j^{t^*}$ where g^v, c_0 (the ciphertext sent by π_i^s) is replayed to $\pi_j^{t^*}$. $\mathcal{B}_{\text{PRF-ODH}}$ sets the computation of ck, k_0 in such sessions to \tilde{ck}, \tilde{k}_0 to ensure consistency. In addition, within the same session $\mathcal{B}_{\text{PRF-ODH}}$ has to potentially simulate the computation of $ck, k_1 \leftarrow \text{KDF}(\tilde{ck}, g^{b^v}, 2)$, where g^b may have been injected by \mathcal{A} . In order to compute this, $\mathcal{B}_{\text{PRF-ODH}}$ queries $\text{ODHv}(\tilde{ck}, g^b, 2)$ to simulate this computation. Thus we have:

$$\text{Adv}(\text{abort}_{win}) \leq \text{Adv}_{\text{KDF}, G, P, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case A, Game 6: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{ab}, 2)$ in π_i^s and its honest partner (if one exists). Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the **prf** security of the key derivation function **KDF**, and thus:

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when **Enc** and **Dec** would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0 replaced in **Game 5**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an **AEAD** challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to **OEnc** or **ODec** 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 \tilde{k}_0 to **Encrypt** or **Decrypt** the first ciphertext). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the **AEAD** challenger's oracles.

Since \tilde{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,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

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:

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

Case A, Game 8: In this game, π_i^s will only set $\text{win} \leftarrow 1$ (and thus cause $\text{abort}_{\text{win}}$ to occur) if \mathcal{A} is able to produce a ciphertext $g^b, c_1 := g^b, \text{AEAD.enc}(\tilde{k}_1, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to both $\text{OEnc}(j, t, m_0, m_1)$ and $\text{ODec}(i, s, c)$ queries directed to an honest partner to π_i^s, π_j^t (if such a session exists) and π_i^s when $\pi_i^s \cdot \varsigma = 1$ from \mathcal{A} (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 $\mathcal{B}_{\text{aead}}$ to break the AEAD security game.

An adversary capable causing $\text{win} \leftarrow 1$ can thus break the aead security of the AEAD scheme. Since \tilde{k}_1 is a uniformly random and independent value (by **Game 6**), this change is indistinguishable. Thus,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

Note that the additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{NK_label} \| ad) \| g^B) \| g^a) \| c_0) \| g^b)$. 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 $\text{abort}_{\text{win}}$ due to π_i^s processing (g^b, c_1) , and we find:

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

We now treat **Case B**. \mathcal{A} in **Case B** cannot cause π_i^s such that $\pi_i^s \cdot \rho = \mathbf{r}$ and $\pi_i^s \cdot \text{fr}_1 = 1$ to set $\text{win} \leftarrow 1$, as $\text{au}^1 = \infty$. If there does not exist some session π_j^t such that $\pi_i^s \cdot \text{pid} = j$ and $\pi_i^s \cdot \text{Td}[1] = \pi_j^t \cdot \text{Te}[1]$, then $\pi_i^s \cdot \text{fr}_1 \leftarrow 0$ and thus win is not set to 1.

In the next case, we know that if $\pi_i^s \cdot \rho = \mathbf{i}$ ($\pi_i^s \cdot \rho = \mathbf{r}$ 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.

$$\text{Adv}_{\text{NK}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CC}} = \text{Adv}(\text{break}_0).$$

Case C, Game 1: In this game we define an abort event $\text{abort}_{\text{coll}}$ 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 $in \neq \hat{in}$ but $\text{H}(in) = \text{H}(\hat{in})$. The simulator $\mathcal{B}_{\text{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\text{Adv}(\text{break}_0) \leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + \text{Adv}(\text{break}_3)$$

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:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P n_S \cdot (\text{Adv}(\text{break}_3)).$$

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

It is clear that

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

Case C.1, Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a ms-PRF-ODH challenger as described in **Case A.1, Game 5**. Thus:

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

Case C.1, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{ab}, 2)$ in π_i^s and its honest partner. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0 replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ (or $\text{ODec}(j, t, c)$) 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the AEAD challenger.

Since \tilde{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,

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_i, \tilde{k}_r are uniformly random and independent values, this change is sound. Thus,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD},\mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{win} \leftarrow 1$. Thus:

$$\begin{aligned} \text{Adv}_{\text{NK},n_P,n_S,\mathcal{A}}^{\text{fACCE,CC.1}} &\leq \text{Adv}_{\text{H},\mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(2 \cdot \text{Adv}_{\text{KDF},\mathcal{B}_{\text{prf}}}^{\text{prf}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF},G,p,\mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + 3 \cdot \text{Adv}_{\text{AEAD},\mathcal{B}_{\text{aead}}}^{\text{aead}} \right) \end{aligned}$$

We now treat **Case C.2**. By the definition of this sub-case, we know that \mathcal{A} has not issued both a $\text{ORevealRandomness}(i, s)$ and a $\text{ORevealRandomness}(j, t)$ query, but has issued a $\text{OCorrupt}(j)$ query. Since $\text{fs} = 2$ and $\text{r1}^\text{r} = \infty$, by Table 3 any adversary that issues a $\text{OCorrupt}(j)$ sets $\pi_i^s.\text{fr}_1 \leftarrow 0$ 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 \tilde{ck}, \tilde{k}_1 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ 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, ζ', 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, $\mathcal{B}_{\text{PRF-ODH}}$ 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_i^s was output by the honest partner π_j^t , we only need to use the private key u in a single case:

$$- ck, k_0 \leftarrow \text{KDF}(ck, g^{uB}, 2)$$

Since the private key B is an honestly generated long-term private key, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$ needs only replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 output by the nn-PRF-ODH challenger. Thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 2$ (i.e. when using the key \tilde{k}_1 replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, m_1)$ (or $\text{ODec}(j, t, c)$) queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 2$ (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the AEAD challenger.

Since \tilde{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,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

We note now that if \mathcal{A} terminates and outputs a tuple (i, s, ζ', b') at the end of the game (where $\zeta' = 2$), 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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 3$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 3$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the AEAD challenger's oracles.

Since \tilde{k}_i, \tilde{k}_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,

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

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 win $\leftarrow 1$. Thus:

$$\begin{aligned} \text{Adv}_{\text{NK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC.2}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(2 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{nn-PRF-ODH}} + \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} \right) \end{aligned}$$

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

E.3 NN Pattern

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

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

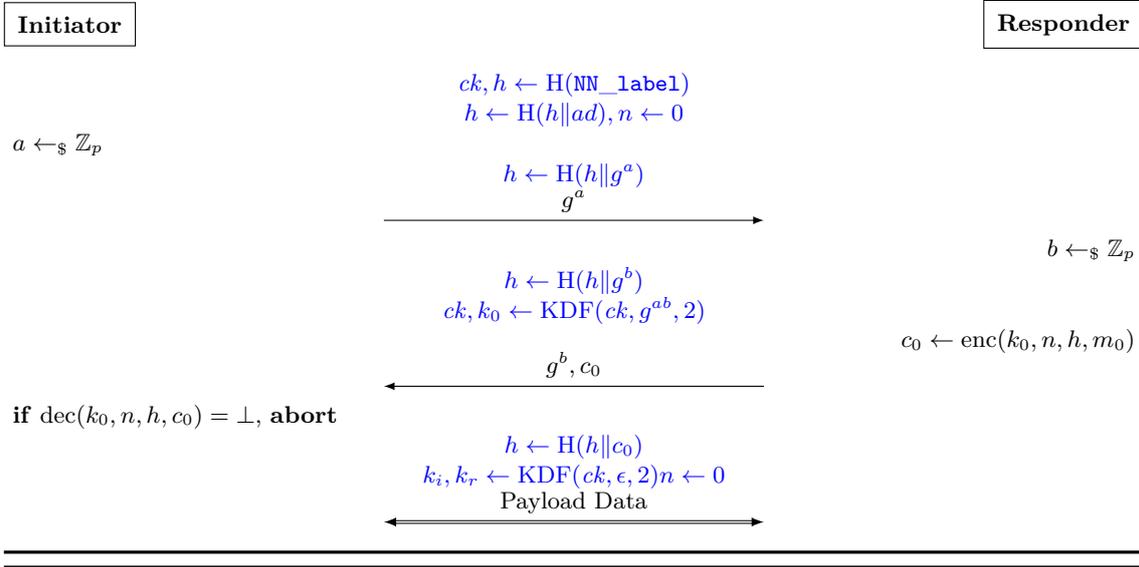


Fig. 10: Noise Pattern NN $\rightarrow \mathbf{e}, \leftarrow \mathbf{e}, \mathbf{ee}$.

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 $\pi_i^s.b$, $\pi_i^s.fr_\zeta = 1 \forall \zeta \geq 1$ and thus $\text{ORevealRandomness}(i, s)$ cannot have been issued by \mathcal{A} as $\mathbf{rl}^1 = \infty$ and $\mathbf{eck} = \infty$. In addition, since

$\text{au}^\tau = \infty$ and $\text{au}^i = \infty$, if \mathcal{A} terminates and outputs a tuple (i, s, ζ', b') where $\pi_i^s.\text{fr}_1 = 1$ then there must exist some session π_j^t such that $\pi_i^s.\text{pid} = j$ 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 $\text{win} \leftarrow 1$ (i.e. breaking authentication). Finally, the honest partner π_j^t (that must exist) $\text{ORevealRandomness}(j, t)$ cannot have been issued by \mathcal{A} as $\text{rl}^\tau = \infty$ and $\text{eck} = \infty$.

Game 0: This is the standard fACCE experiment.

$$\text{Adv}_{\text{NN}, n_P, n_S, \mathcal{A}}^{\text{fACCE}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P n_S \cdot (\text{Adv}(\text{break}_3)).$$

Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a nn-PRF-ODH challenger in the following way: $\mathcal{B}_{\text{PRF-ODH}}$ 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_i^s was output by the honest partner π_j^t , we only need to use the private key u in a single case:

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

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

$$\text{Adv}(\text{break}_3) \leq \text{Adv}_{\text{KDF}, G, P, \mathcal{B}_{\text{PRF-ODH}}}^{\text{nn-PRF-ODH}} + \text{Adv}(\text{break}_4)$$

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

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

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 $\text{OEnc}(i, s, m_0, \tilde{m}_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0 replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, \tilde{m}_1)$ (or $\text{ODec}(j, t, c)$) 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the AEAD challenger.

Since \tilde{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,

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

We note now that if \mathcal{A} terminates and outputs a tuple (i, s, ζ', b') at the end of the game (where $\zeta' = 1$), 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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the keys \tilde{k}_i and \tilde{k}_r replaced in **Game 5**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ (or $\text{ODec}(j, t, c)$) 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 \tilde{k}_i or \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the AEAD challenger.

Since \tilde{k}_i, \tilde{k}_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,

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

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:

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

E.4 XN Pattern

Theorem 5. *The Noise protocol XN is flexible-ACCE-secure protocol with authentication levels $\mathbf{au} = (3, \infty)$, forward-secrecy $\mathbf{fs} = 2$, KCI resistance $\mathbf{kc} = (3, \infty)$, one-way-randomness-security $\mathbf{r1} = (\infty, 3)$, eCK security $\mathbf{eck} = \infty$, replay resistance $\mathbf{rp} = (2, 0)$. That is, for an adversary \mathcal{A} against the flexible ACCE security game (defined in Section 5) one can define efficient adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of H, $\mathcal{B}_{\text{PRF-ODH}}$ against the sym-ms-PRF-ODH and nn-PRF-ODH assumptions with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the PRF security of KDF with:*

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

Proof.

We split the analysis into the following three cases:

- **Case A:** \mathcal{A} outputs (i, s, ς', b') (where $\pi_i^s.\rho = \mathbf{i}$) and \mathcal{A} causes π_i^s to set $\text{win} \leftarrow 1$ 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 $\pi_i^s.\rho = \mathbf{r}$) and \mathcal{A} causes π_i^s to set $\text{win} \leftarrow 1$ 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 $\text{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_0 (if $\pi_i^s.\rho = \mathbf{i}$) or when π_i^s processes the ciphertext c_1, c_2 (if $\pi_i^s.\rho = \mathbf{r}$).

We begin by treating **Case A**. \mathcal{A} in **Case A** cannot cause π_i^s such that $\pi_i^s.\rho = \mathbf{i}$ and $\pi_i^s.\text{fr}_2 = 1$ to set $\text{win} \leftarrow 1$, as $\mathbf{au}^{\mathbf{r}} = \infty$. There must exist some session π_j^t such that $\pi_i^s.\text{pid} = j$ and $\pi_i^s.T_d[1] = \pi_j^t.T_e[1]$, otherwise $\pi_i^s.\text{fr}_1 \leftarrow 0$ and thus win is not set to 1.

We now turn to treating **Case B**. In order for π_i^s to set $\text{win} \leftarrow 1$ in **Case B**, $\pi_i^s.\text{fr}_3 = 1$ and thus $\text{OCorrupt}(\pi_i^s.\text{pid})$ cannot yet have been issued by \mathcal{A} as $\mathbf{kc}^{\mathbf{r}} = \infty > \mathbf{au}^{\mathbf{i}} = 3$, and π_i^s will output $\varsigma = 3$ when decrypting c_1, c_2 . Similarly, \mathcal{A} also cannot have issued a $\text{ORevealRandomness}(i, s)$ query as $\mathbf{r1}^{\mathbf{r}} = 3$ and $\mathbf{eck} = \infty$.

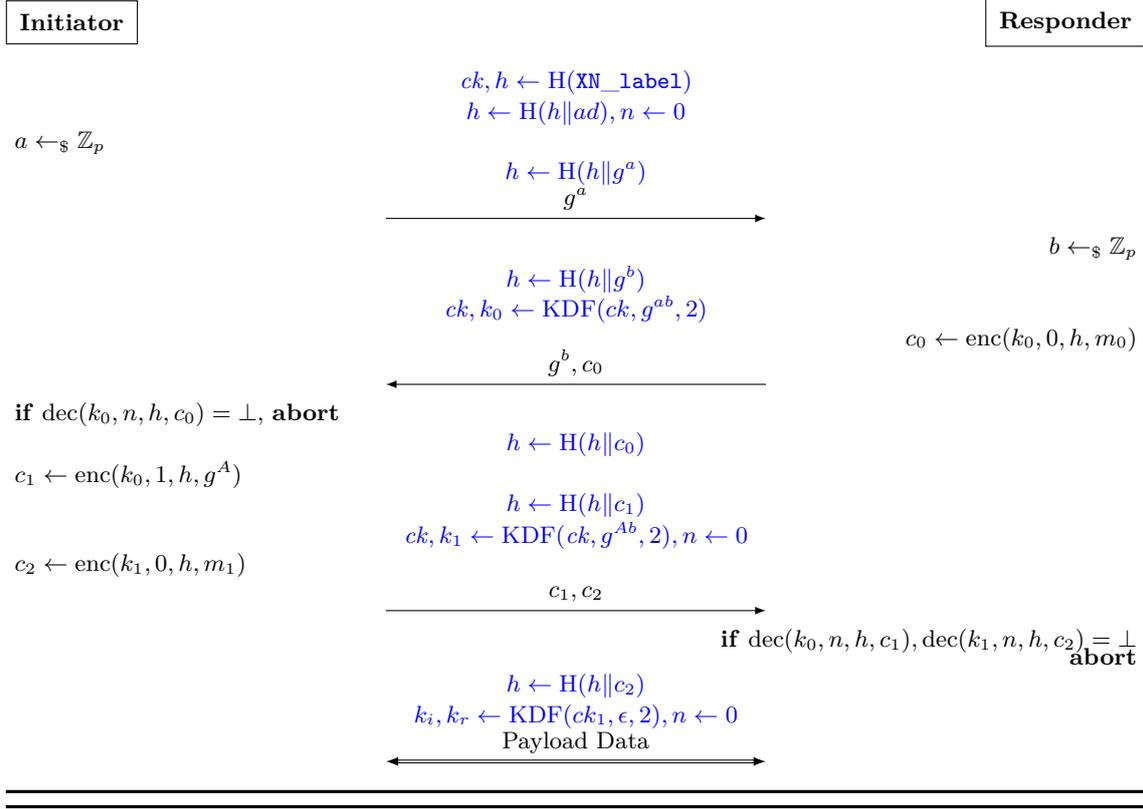


Fig. 11: Noise Pattern $\text{XN} \rightarrow \mathbf{e}, \leftarrow \mathbf{e}, \mathbf{ee}, \rightarrow \mathbf{s}, \mathbf{se}$.

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

$$\text{Adv}_{\text{XN}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CB}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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 $\pi_i^s.\text{pid} \neq j$. Thus:

$$\text{Adv}(\text{break}_2) = n_P \cdot (\text{Adv}(\text{break}_3)).$$

Case B, Game 4: In this game, we define an abort event $\text{abort}_{\text{win}}$ if the session π_i^s sets the status $\text{win} \leftarrow 1$ when π_i^s processes the ciphertext c_1, c_2 (i.e. when π_i^s outputs $\zeta = 3$). Note that the behavior of a test session π_i^s with $\pi_i^s.\rho = \mathbf{r}$ and $\pi_i^s.b_3 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathbf{r}$ 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 $\zeta = 3$ (i.e. after sending the ciphertext g^b, c_0 , before it enters stage $\zeta = 3$ 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 $\text{win} = 1$, 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 $\text{win} \leftarrow 1$ 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 $\text{Adv}(\text{break}_4)$, (the advantage of \mathcal{A} in guessing $\pi_i^s.b_2$, which we refer to as **Case B.1**), and
- We separately bound $\text{Adv}(\text{abort}_{\text{win}})$ (which we refer to as **Case B.2**).

Thus:

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

Case B.1, Game 4: In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a ns-PRF-ODH challenger in the following way: $\mathcal{B}_{\text{PRF-ODH}}$ initializes a ns-PRF-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:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_0 replaced in **Game 4**). We do so by constructing an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 4** except responding to $\text{OEnc}(i, s, m_0, m_1)$ (or $\text{ODec}(j, t, c)$) 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ instead forwards the queries to the AEAD challenger.

Since \tilde{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,

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

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:

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}}$$

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

Case B.2, Game 5: In this game, we replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-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 j of the session π_i^s . Thus, $\mathcal{B}_{\text{PRF-ODH}}$ initializes a sym-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 $\text{OCorrupt}(j)$ query. However, $\mathcal{B}_{\text{PRF-ODH}}$ 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: $ck, k_1 \leftarrow \text{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHu oracle provided by the sym-ms-PRF-ODH challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. Similarly, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $ck, k_0 := \text{KDF}(ck, g^{av})$, where g^a may have been contributed by \mathcal{A} . In this case, in order to compute ck, k_0 , $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHv oracle provided by the sym-ms-PRF-ODH challenger, specifically querying $\text{ODHv}(ck, g^a)$, which will output $\text{KDF}(ck, g^{av})$. After processing c_1, c_2 , π_i^s will output stage $\varsigma = 3$, and so \mathcal{A} cannot issue a $\text{OCorrupt}(j)$ query *before* π_i^s processes ciphertext c_1, c_2 . In addition, since in **Case B**, we abort if $\text{abort}_{\text{win}}$ 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** $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{OCorrupt}(j)$ query. In addition, since $\text{rl}^f = 3$, $\mathcal{B}_{\text{PRF-ODH}}$ also never has to answer a $\text{ORevealRandomness}(i, s)$ query. Thus we have:

$$\text{Adv}(\text{abort}_{\text{win}}) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case B.2, Game 6: In this game, π_i^s will only set $\text{win} \leftarrow 1$ 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 $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{ODec}(i, s, c)$ queries from \mathcal{A} when π_i^s would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ instead queries $\text{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable causing $\text{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since \tilde{k}_1 is a uniformly random and independent value, this change is sound. Thus,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

Note that the additional-data field of c_2 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{XN_label} \| ad) \| g^a) \| g^b) \| c_0) \| c_1)$. 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 $\text{abort}_{\text{win}}$ due to π_i^s processing (c_1, c_2) . Thus:

$$\begin{aligned} \text{Adv}_{\text{XN}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CB}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S \cdot \left(2 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ns-PRF-ODH}} \right) \end{aligned}$$

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 $\pi_i^s.\rho = \mathbf{i}$ or when π_i^s outputs $\varsigma = 3$ (if $\pi_i^s.\rho = \mathbf{r}$). We can now treat **Case C**.

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

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

Case C, Game 1: In this game we define an abort event $\text{abort}_{\text{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \text{H}(in))$,

$(\hat{in}, H(\hat{in}))$ such that $in \neq \hat{in}$ but $H(in) = H(\hat{in})$. The simulator $\mathcal{B}_{\text{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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 $\zeta = 2$ when processing the ciphertext g^b, c_0 (if $\pi_i^s \cdot \rho = \mathbf{i}$) or when π_i^s outputs $\zeta = 3$ when processing the ciphertext c_1, c_2 (if $\pi_i^s \cdot \rho = \mathbf{r}$). Note that by **Case A** and **Case B**, there *must* exist such an honest partner. Thus:

$$\text{Adv}(\text{break}_2) = n_P n_S \cdot (\text{Adv}(\text{break}_3)).$$

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

It is clear that

$$\text{Adv}_{\mathbb{XN}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC}} \leq \max(\text{Adv}_{\mathbb{XN}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC.1}}, \text{Adv}_{\mathbb{XN}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC.2}})$$

Case C.1, Game 4: In this game, we replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-ms-PRF-ODH challenger as described in **Case B Game 5**. Thus:

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

Case C.1, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, \epsilon, 2)$ used to compute $k_1, k_r := \text{KDF}(\tilde{ck}, \epsilon, 2)$. Since, by **Game 4**, \tilde{ck} is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values k_1, k_r with uniformly random values \tilde{k}_1, \tilde{k}_r . Distinguishing this change implies an algorithm breaking the prf security of the KDF scheme, and thus:

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

Case C.1, Game 6: In this game, the challenger flips a bit \bar{b} and uses \bar{b} instead of $\pi_i^s \cdot b_3$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 3$ (i.e. when using the key \tilde{k}_1) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 4** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 3$ (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_1 is a uniformly random and independent value, this change is sound. Thus,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

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

Case C.1, Game 7: In this game, the challenger flips a bit \bar{b}' and uses \bar{b}' instead of $\pi_i^s \cdot b_4$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output

$\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_i, \tilde{k}_r are uniformly random and independent values, this change is sound. Thus,

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

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 $\text{win} \leftarrow 1$. Thus:

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

We now treat **Case C.2, Game 4**: In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ 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, $\mathcal{B}_{\text{PRF-ODH}}$ initializes a nn-PRF-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:

$$- ck, k_1 \leftarrow \text{KDF}(ck, g^{uA}, 2)$$

Since the private key A is an honestly generated long-term private key, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$ needs only replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 output by the nn-PRF-ODH challenger. Thus:

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

Case C.2, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{aB}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{aB}, 2)$. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the **prf** security of the key derivation function KDF, and thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_0) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ 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 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,

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key \tilde{k}_1) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_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,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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 $\text{win} \leftarrow 1$. Thus:

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

E.5 NX Pattern

Theorem 6. *The Noise protocol NX is flexible-ACCE-secure protocol with authentication levels $\mathbf{au} = (\infty, 2)$, forward-secrecy $\mathbf{fs} = 2$, KCI resistance $\mathbf{kc} = (\infty, 2)$, one-way-randomness-security $\mathbf{r1} = (2, \infty)$, eCK security $\mathbf{eck} = \infty$ replay resistance $\mathbf{rp} = (2, 0)$. That is, for an adversary \mathcal{A} against the flexible ACCE security game (defined in Section 5) one can define efficient adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of H, $\mathcal{B}_{\text{PRF-ODH}}$ against the sym-ms-PRF-ODH and nn-PRF-ODH assumptions with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the PRF security of KDF*

with:

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

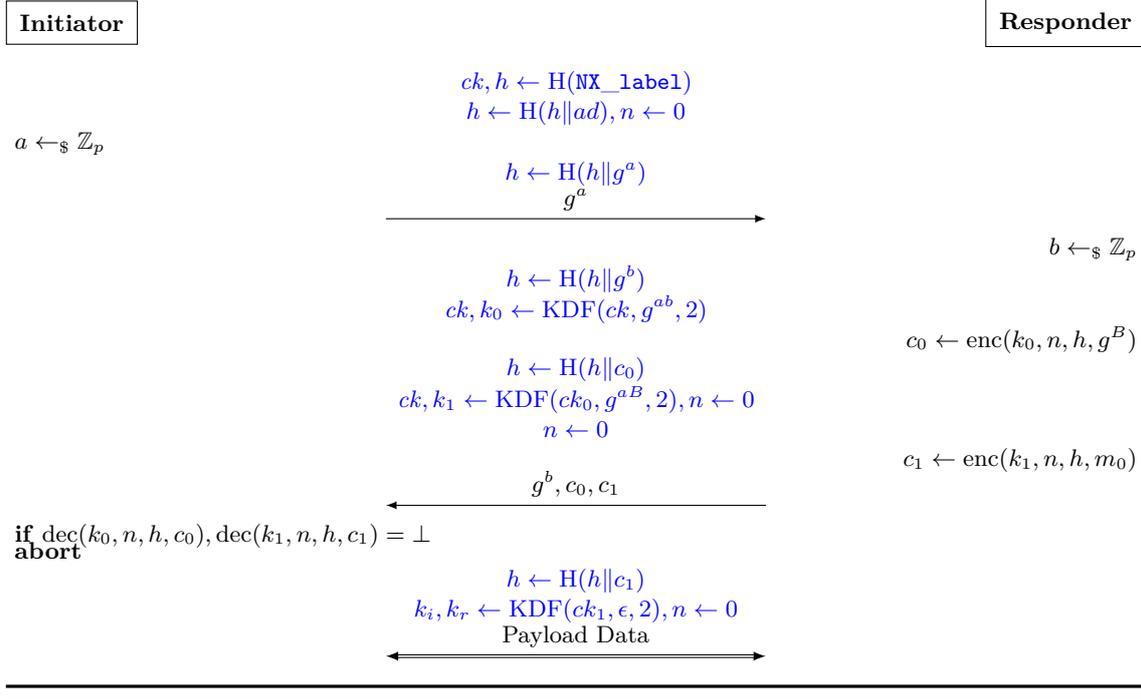


Fig. 12: Noise Pattern NX \rightarrow e, \leftarrow e, ee, s, es.

Proof.

We split the analysis into the following three cases:

- **Case A:** \mathcal{A} outputs (i, s, ζ', b') (where $\pi_i^s \cdot \rho = i$) and \mathcal{A} causes π_i^s to set $\text{win} \leftarrow 1$ when π_i^s processes the second ciphertext g^b, c_0, c_1 (i.e when π_i^s outputs $\zeta = 2$ during a decryption call).
- **Case B:** \mathcal{A} outputs (i, s, ζ', b') (where $\pi_i^s \cdot \rho = r$) and \mathcal{A} causes π_i^s to set $\text{win} \leftarrow 1$ when π_i^s processes the first ciphertext g^a (i.e when π_i^s outputs $\zeta = 2$ during a decryption call).
- **Case C:** \mathcal{A} outputs (i, s, ζ', b') and \mathcal{A} does not cause π_i^s to set $\text{win} \leftarrow 1$ when π_i^s outputs $\zeta = 2$ during a decryption call (if $\pi_i^s \cdot \rho = i$) or when π_i^s outputs $\zeta = 1$ during a decryption call (if $\pi_i^s \cdot \rho = r$).

We begin by treating **Case A**. In order for π_i^s to set $\text{win} \leftarrow 1$ in **Case A**, $\pi_i^s \cdot \text{fr}_2 = 1$ and thus $\text{OCorrupt}(\pi_i^s \cdot \text{pid})$ cannot yet have been issued by \mathcal{A} as $\text{kc}^r = 2 = \text{au}^r = 2$, and π_i^s will output $\zeta = 2$ when decrypting g^b, c_0, c_1 . Similarly, \mathcal{A} also cannot have issued a $\text{ORevealRandomness}(i, s)$ query as $\text{r1}^r = \infty$ and $\text{eck} = \infty$.

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

$$\text{Adv}_{\text{NX}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CA}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

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

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:

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

Case A, Game 4: In this game, we define an abort event $abort_{\text{win}}$ if the session π_i^s sets the status $\text{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_0, c_1 (i.e. when π_i^s outputs $\zeta = 2$). Note that the behavior of a test session π_i^s with $\pi_i^s.\rho = 1$ and $\pi_i^s.b_2 = 1$ does not differ from a session with $\pi_i^s.\rho = 1$ 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 $\zeta = 2$ (i.e. after sending the ciphertext g^a , before it enters stage $\zeta = 2$ 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 $\text{win} = 1$, thus $\text{Adv}(break_4) = 0$. 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 $\text{win} \leftarrow 1$ 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 $\text{Adv}(abort_{\text{win}})$. Thus:

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

Case A, Game 5: In this game, we replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-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, ζ', 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, $\mathcal{B}_{\text{PRF-ODH}}$ initializes a sym-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 $\text{OCorrupt}(j)$ query. However, $\mathcal{B}_{\text{PRF-ODH}}$ must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol **NX**, the long-term private keys are used to compute the following:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHu oracle provided by the sym-ms-PRF-ODH challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. Similarly, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $ck, k_0 := \text{KDF}(ck, g^{av})$, where g^a may have been contributed by \mathcal{A} . In this case, in order to compute ck, k_0 , $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHv oracle provided by the sym-ms-PRF-ODH challenger, specifically querying $\text{ODHv}(ck, g^a)$, which will output $\text{KDF}(ck, g^{av})$. We note that $\text{au}^r = 1$, and while processing g^b, c_0, c_1 , π_i^s will output $\zeta = 2$, and so \mathcal{A} cannot issue a $\text{OCorrupt}(j)$ 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, ζ', b')), by **Game 4** $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{OCorrupt}(j)$ query. Finally, since $r_1^1 = 2$, $\mathcal{B}_{\text{PRF-ODH}}$ also never has to answer a $\text{ORevealRandomness}(i, s)$ query. Thus we have:

$$\text{Adv}(\text{abort}_{win}) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case A, Game 6: In this game, π_i^s will only set $\text{win} \leftarrow 1$ if \mathcal{A} is able to produce a ciphertext $g^b, c_1, c_2 := g^b, c_1, \text{AEAD.enc}(\tilde{k}_1, n, h, m)$ that decrypts without error. In session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{ODec}(i, s, c)$ queries from \mathcal{A} when π_i^s would output $\zeta = 2$ (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ instead queries $\text{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable causing $\text{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since \tilde{k}_1 is a uniformly random and independent value, this change is sound. Thus,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

Note that the additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{NX_label} \| ad) \| g^a) \| g^b) \| c_0)$. 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:

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

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 $\pi_i^s \cdot \rho = r$ and $\pi_i^s \cdot fr_1 = 1$ to set $\text{win} \leftarrow 1$, as $\text{au}^1 = \infty$. There must exist some session π_j^t such that $\pi_i^s \cdot pid = j$ and $\pi_i^s \cdot T_d[1] = \pi_j^t \cdot T_e[1]$, otherwise $\pi_i^s \cdot fr_1 \leftarrow 0$ 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.

$$\text{Adv}_{\text{NX}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC}} = \text{Adv}(\text{break}_0).$$

Case C, Game 1: In this game we define an abort event $\text{abort}_{\text{coll}}$ 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 $in \neq \hat{in}$ but $\text{H}(in) = \text{H}(\hat{in})$. The simulator $\mathcal{B}_{\text{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\text{Adv}(\text{break}_0) \leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + \text{Adv}(\text{break}_3)$$

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P n_S \cdot (\text{Adv}(\text{break}_3)).$$

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 $\text{OCorrupt}(j)$ query and an $\text{ORevealRandomness}(i, s)$ query.

- **Case C.2:** \mathcal{A} has not issued both a $\text{ORevealRandomness}(i, s)$ and a $\text{ORevealRandomness}(j, t)$ query. Note that if \mathcal{A} issues a $\text{OCorrupt}(j)$ query, then $\pi_i^s \cdot \text{fr}_1 \leftarrow 0$, and thus \mathcal{A} has no advantage in outputting $(i, s, 1, b')$. If \mathcal{A} did not issue a $\text{OCorrupt}(j)$ 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 \tilde{ck}, \tilde{k}_1 in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-ms-PRF-ODH challenger as described in **Case A, Game 5**. Thus:

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

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

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

Case C.1, Game 6: In this game, the challenger flips a bit \bar{b} and uses \bar{b} instead of $\pi_i^s \cdot b_2$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_1) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 k_1 is a uniformly random and independent value, and the bit \bar{b} has the same distribution as $\pi_i^s \cdot b_2$, this change is sound. Thus,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

In **Case C.1, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s \cdot b_\varsigma$, 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 $\pi_i^s \cdot b_3$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_r are uniformly random and independent values, and the bit \bar{b}' has the same distribution as $\pi_i^s \cdot b_3$, this change is sound. Thus,

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

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

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

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 \tilde{ck}, \tilde{k}_0 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ 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, $\mathcal{B}_{\text{PRF-ODH}}$ 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:

$$- ck, k_1 \leftarrow \text{KDF}(ck, g^{uB}, 2)$$

Since the private key B is an honestly generated long-term private key, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$ needs only replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 output by the nn-PRF-ODH challenger. Thus:

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

Case C.2, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{aB}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{aB}, 2)$. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ 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 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,

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_1) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

In **Case C.2, Game 8**, the behavior of π_i^s is independent of the test bit $\pi_i^s \cdot b_\zeta$, where $\zeta = 2$ 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 $\pi_i^s \cdot b_3$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 3$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 3$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_r are uniformly random and independent values, and the bit \bar{b}^* has the same distribution as $\pi_i^s \cdot b_3$, this change is sound. Thus,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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

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

E.6 X pattern

Theorem 7. *The Noise protocol X is flexible-ACCE-secure protocol with authentication levels $\mathbf{au} = (1, \infty)$, forward-secrecy $\mathbf{fs} = \infty$, KCI resistance $\mathbf{kc} = (\infty, \infty)$, randomness-security $\mathbf{rl} = (1, \infty)$, eCK security $\mathbf{eck} = 1$, replay resistance $\mathbf{rp} = (\infty, \infty)$. That is, for an adversary \mathcal{A} against the flexible ACCE security game (defined in section 5) one can define efficient adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of H, $\mathcal{B}_{\text{PRF-ODH}}$ against the sym-ms-PRF-ODH and nn-PRF-ODH assumptions with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the PRF security of KDF with:*

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

Proof.

We split the analysis into the following two cases:

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

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

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

$$\text{Adv}_{\text{X}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CA}} = \text{Adv}(\text{break}_0).$$

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

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

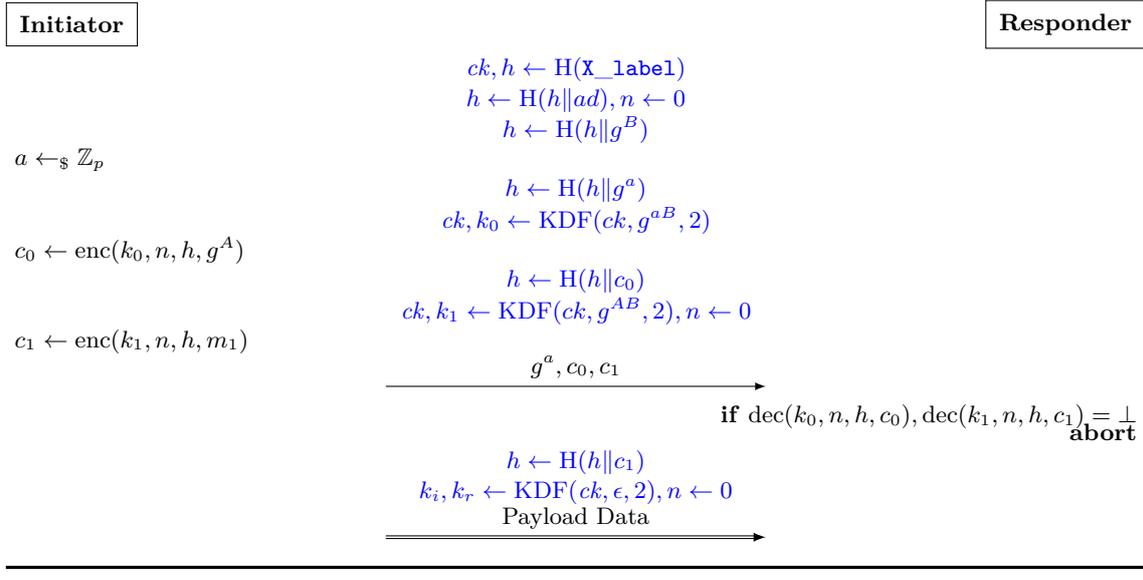


Fig. 13: Noise Pattern $\mathbf{X} : \leftarrow \mathbf{s}, \dots, \rightarrow \mathbf{e}, \mathbf{es}, \mathbf{s}, \mathbf{ss}$.

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S (\text{Adv}(\text{break}_2)).$$

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 $\pi_i^s.\text{pid} \neq j$. Thus:

$$\text{Adv}(\text{break}_2) = n_P (\text{Adv}(\text{break}_3)).$$

Case A, Game 4: In this game, we define an abort event abort_{win} if the session π_i^s sets the status $\text{win} \leftarrow 1$ when π_i^s processes the ciphertext g^a, c_0, c_1 (i.e when π_i^s outputs $\zeta = 1$). Note that the behavior of a test session π_i^s with $\pi_i^s.\rho = \mathbf{r}$ and $\pi_i^s.b_1 = 1$ does not differ from a session with $\pi_i^s.\rho = \mathbf{r}$ 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 $\zeta = 1$ (i.e. outputs $\zeta = 1$ 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 $\text{win} = 1$, thus $\text{Adv}(\text{break}_4) = 0$. 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 $\text{win} \leftarrow 1$ 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 $\text{Adv}(\text{abort}_{win})$. Thus:

$$\text{Adv}(\text{break}_3) \leq \text{Adv}(\text{abort}_{win}).$$

Case A, Game 5: In this game, we replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 . We do so by defining an algorithm $\mathcal{B}_{\text{PRF-ODH}}$ 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, ζ', 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, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\text{OCorrupt}(j)$ or $\text{OCorrupt}(i)$ queries, as $\mathbf{fs} = \infty, \mathbf{au} = (1, \infty)$ and $\mathbf{kc} = (\infty, \infty)$. However, $\mathcal{B}_{\text{PRF-ODH}}$ 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 \mathbf{X} , the long-term private keys are used to compute the following:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

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

$$\text{Adv}(\text{abort}_{\text{win}}) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-mm-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case A, Game 6: In this game, π_i^s will only set $\text{win} \leftarrow 1$ if \mathcal{A} is able to produce a ciphertext $g^a, c_0, c_1 := g^a, c_0, \text{AEAD.enc}(\tilde{k}_1, n, h, m)$ that decrypts without error. In this game, we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{ODec}(j, t, c)$ queries directed to π_i^s when $\zeta = 0$ from \mathcal{A} (i.e. when using the key \tilde{k}_1). Instead, $\mathcal{B}_{\text{aead}}$ simply forwards these queries to the AEAD challenger.

An adversary capable of causing $\text{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Note that the additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{X_label} \| ad) \| g^B) \| g^a) \| c_0$. 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 $\text{abort}_{\text{win}}$ due to π_i^s processing (g^a, c_0, c_1) . Thus:

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}}$$

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

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 $\pi_i^s.\rho = r$ 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 $\pi_i^s.\rho = i$ 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.

$$\text{Adv}_{\text{X}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CB}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S (\text{Adv}(\text{break}_1)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P n_S \cdot (\text{Adv}(\text{break}_3)).$$

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 $\text{OCorrupt}(i)$ and a $\text{OCorrupt}(j)$ query. Note that if \mathcal{A} issues a $\text{ORevealRandomness}(j, t)$ query at the beginning of the game, then $\pi_i^s \cdot \text{fr}_1 \leftarrow 0$, and thus \mathcal{A} has no advantage in outputting $(i, s, 1, b')$. If \mathcal{A} did not issue a $\text{ORevealRandomness}(j, t)$ query, then the security analysis reverts to **Case B.2**.
- **Case B.2:** \mathcal{A} has not issued both a $\text{OCorrupt}(i)$ query and an $\text{ORevealRandomness}(j, t)$ query.

Case B.1, Game 4: In this game, we replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 . We do so defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-mm-PRF-ODH challenger in the same way as **Case A, Game 5**. Thus we have:

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

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

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

Case B.1, Game 6: In this game, the challenger flips a bit \bar{b} and uses \bar{b} instead of $\pi_i^s \cdot b_1$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_1) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_1 is a uniformly random and independent value, and the bit \bar{b} has the same distribution as $\pi_i^s \cdot b_1$, this change is sound. Thus,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

In **Case B.1, Game 6**, the behavior of π_i^s is independent of the test bit $\pi_i^s \cdot b_\varsigma$, 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 $\pi_i^s \cdot b_2$ when responding to $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_i, \tilde{k}_r are uniformly random and independent values, and the bit \bar{b}' has the same distribution as $\pi_i^s \cdot b_2$, this change is sound. Thus,

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

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

$$\begin{aligned} \text{Adv}_{X,n_P,n_S,\mathcal{A}}^{\text{fACCE,CB.1}} &\leq \text{Adv}_{H,\mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(\text{Adv}_{\text{KDF},\mathcal{B}_{\text{prf}}}^{\text{prf}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF},G,p,\mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-mm-PRF-ODH}} + 2 \cdot \text{Adv}_{\text{AEAD},\mathcal{B}_{\text{aead}}}^{\text{aead}} \right) \end{aligned}$$

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 \tilde{ck}, \tilde{k}_0 . We do so by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a ms-PRF-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 $\text{ORevealRandomness}(i, s)$ query, nor a $\text{OCorrupt}(j)$ 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, $\mathcal{B}_{\text{PRF-ODH}}$ 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 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: $ck, k_1 \leftarrow \text{KDF}(ck, g^{uB}, 2)$
- In sessions where the party acts as the responder: $ck, k_0 \leftarrow \text{KDF}(ck, g^{au}, 2)$, $ck, k_1 \leftarrow \text{KDF}(ck, g^{Au}, 2)$

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

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

$$\text{Adv}(\text{break}_4) \leq \text{Adv}_{\text{KDF},G,p,\mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case B.2, Game 6: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{AB}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{AB}, 2)$. Since, by Game 5, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the PRF security of the key derivation function KDF, and thus:

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

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

$$\text{Adv}(\text{break}_6) \leq \text{Adv}_{\text{KDF},\mathcal{B}_{\text{prf}}}^{\text{prf}} + \text{Adv}(\text{break}_7)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_1) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 7**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 9** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_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,

$$\text{Adv}(\text{break}_9) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_{10})$$

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 $\text{win} \leftarrow 1$. Thus:

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

E.7 XX pattern

Theorem 8. *The Noise protocol XX is flexible-ACCE-secure protocol with authentication levels $\mathbf{au} = (3, 2)$, forward-secrecy $\mathbf{fs} = 2$, KCI resistance $\mathbf{kc} = (3, 2)$, randomness-security $\mathbf{r1} = (2, 3)$, eCK security $\mathbf{eck} = \infty$, replay resistance $\mathbf{rp} = (2, 0)$. That is, for an adversary \mathcal{A} against the flexible ACCE security game (defined in Section 5) one can define efficient adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of H , $\mathcal{B}_{\text{PRF-ODH}}$ against the nn-PRF-ODH and sym-ms-PRF-ODH assumptions with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the PRF security of KDF with: More precisely:*

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

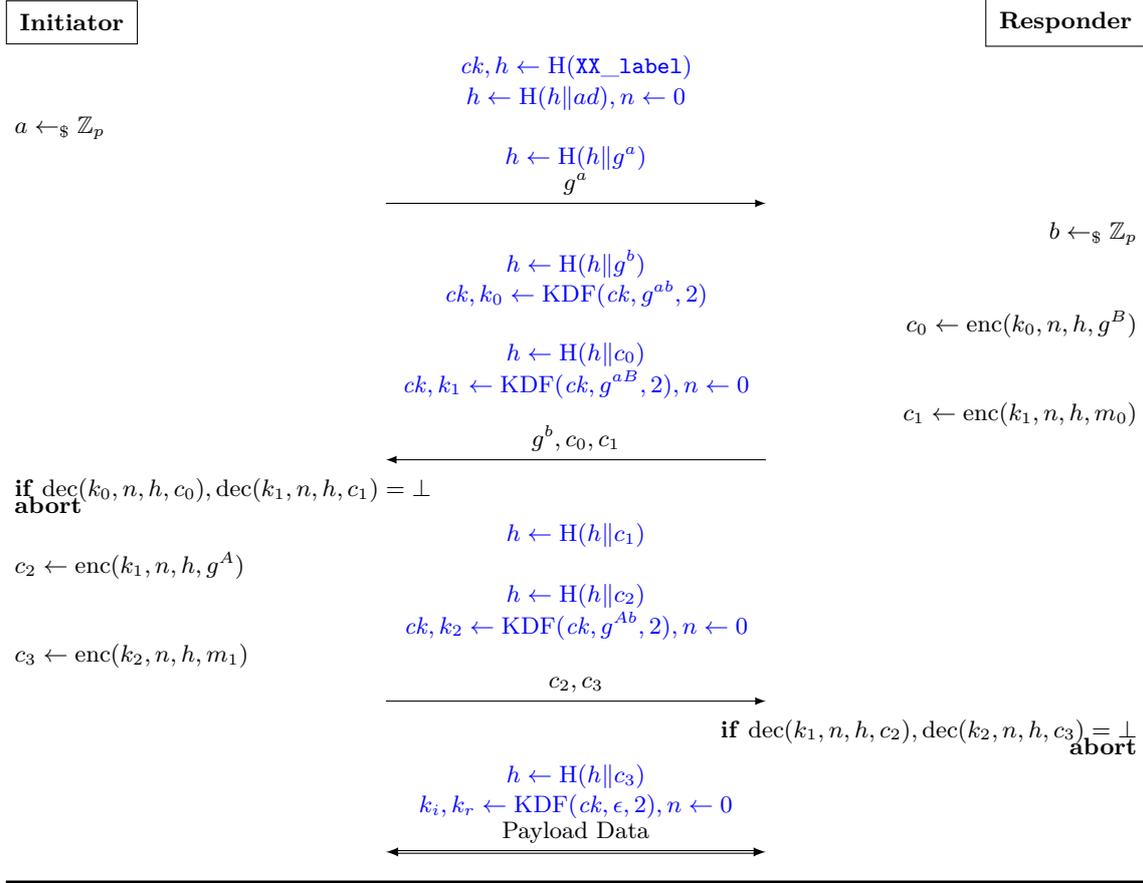


Fig. 14: Noise Pattern $\text{XX} : \rightarrow \mathbf{e}, \leftarrow \mathbf{e}, \mathbf{ee}, \mathbf{s}, \mathbf{es}, \rightarrow \mathbf{s}, \mathbf{se}$.

Proof.

We split the analysis into the following three cases:

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

We begin by treating **Case A**. In order for π_i^s to set $\text{win} \leftarrow 1$ in **Case A**, $\pi_i^s \cdot \text{fr}_1 = 1$ and thus $\text{OCorrupt}(\pi_i^s \cdot \text{pid})$ cannot have been issued by \mathcal{A} as $\text{au}^{\mathbf{x}} = 2$ and π_i^s will output $\pi_i^s \cdot \zeta = 1$ after it processes the ciphertext g^b, c_0, c_1 . Also, \mathcal{A} cannot have issued a $\text{ORevealRandomness}(i, s)$ query as $\text{r1}^{\mathbf{i}} = 2$ and $\text{eck} = \infty$.

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

$$\text{Adv}_{\text{XX}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CA}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P \cdot (\text{Adv}(\text{break}_3)).$$

Case A, Game 4: In this game, we define an abort event abort_{win} if the session π_i^s sets the status $\text{win} \leftarrow 1$ 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} .

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

Case A, Game 5: By the definition of this case, and the Noise Protocol XX we know that $\text{r1}^i = 2$ and $\text{kc}^i = 3 > \text{au}^r = 2$ 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 \tilde{ck}, \tilde{k}_1 . We do so by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a **sym-ms-PRF-ODH** challenger in the following way:

By the definition of this case (**Case A**), we know that \mathcal{A} has not issued a **ORevealRandomness**(i, s) query, nor a **OCorrupt**(j) query. Thus, $\mathcal{B}_{\text{PRF-ODH}}$ initializes a **sym-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. However, we must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol XX, the long-term private keys are used to compute the following:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

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

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

$$\text{Adv}(\text{break}_4) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case A, Game 6: In this game, π_i^s will only set $\text{win} \leftarrow 1$ if \mathcal{A} is able to produce a ciphertext $g^b, c_0, c_1 := g^b, c_0, \text{AEAD.enc}(\tilde{k}_1, n, h, m)$ that decrypts without error. In this game, we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an **AEAD** challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to **ODec**(j, t, c) queries directed to π_i^s when π_i^s would output $\varsigma = 1$ from \mathcal{A} (i.e. when using the key \tilde{k}_1). Instead, $\mathcal{B}_{\text{aead}}$ simply forwards these queries to the **AEAD** challenger.

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

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

Note that the additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{XX_label} \| ad) \| g^a) \| g^b) \| c_0$. 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 $\text{abort}_{win} = 0$ due to π_i^s processing (g^b, c_0, c_1) is 0. Thus:

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

Now, we treat **Case B**. In order for π_i^s to set $\text{win} \leftarrow 1$ in **Case B**, $\pi_i^s.\text{fr}_3 = 1$ and thus $\text{OCorrupt}(\pi_i^s.\text{pid})$ cannot have yet been issued by \mathcal{A} as $\text{au}^1 = 3$ and π_i^s will only output $\zeta = 3$ after decrypting the ciphertext c_2, c_3 . Also, \mathcal{A} cannot have issued a $\text{ORevealRandomness}(i, s)$ query as $\text{r1}^\tau = 3$ and $\text{eck} = \infty$.

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

$$\text{Adv}_{\text{XX}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CB}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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

$$\text{Adv}(\text{break}_2) = n_P \cdot (\text{Adv}(\text{break}_3)).$$

Case B, Game 4: In this game, we define an abort event $\text{abort}_{\text{win}}$ if the session π_i^s sets the status $\text{win} \leftarrow 1$ 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 $\text{abort}_{\text{win}}$.

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

Case B, Game 5: In this game, we replace the computation of ck, k_2 with uniformly random and independent values \tilde{ck}, \tilde{k}_2 . We do so by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-ms-PRF-ODH challenger in the following way:

By the definition of this case (**Case B**), we know that \mathcal{A} has not issued a $\text{ORevealRandomness}(i, s)$ query, nor a $\text{OCorrupt}(j)$ query. Thus, $\mathcal{B}_{\text{PRF-ODH}}$ initializes a sym-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. However, we must account for all sessions t such that party j must use the private key for computations. In the Noise Protocol **XX**, the long-term private keys are used to compute the following:

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

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

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

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

$$\text{Adv}(\text{break}_4) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}(\text{break}_5)$$

Case B, Game 6: In this game, π_i^s will only set $\text{win} \leftarrow 1$ if \mathcal{A} is able to produce a ciphertext $c_2, c_3 := c_2 \text{AEAD.enc}(\tilde{k}_2, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts

with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{ODec}(i, s, c)$ queries directed to π_i^s when π_i^s would output $\varsigma = 3$ from \mathcal{A} (i.e. when using the key k_2). $\mathcal{B}_{\text{aead}}$ instead queries $\text{Dec}(n, h, c)$ to the AEAD challenger's oracles.

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

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

Note that the additional-data field of c_3 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{XX_label})\|ad)\|g^a)\|g^b)\|c_0)\|c_1)$. 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 $\text{abort}_{\text{win}} = 1$ due to π_i^s processing (c_2, c_3) is 0. Thus:

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

Finally, we treat **Case C**.

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

$$\text{Adv}_{\text{XX}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CC}} = \text{Adv}(\text{break}_0).$$

Case C, Game 1: In this game we define an abort event $\text{abort}_{\text{coll}}$ 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 $in \neq \hat{in}$ but $\text{H}(in) = \text{H}(\hat{in})$. The simulator $\mathcal{B}_{\text{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\text{Adv}(\text{break}_0) \leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + \text{Adv}(\text{break}_3)$$

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:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P n_S \cdot (\text{Adv}(\text{break}_3)).$$

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 $\text{OCorrupt}(i)$ query and an $\text{ORevealRandomness}(j, t)$ query. Note that if \mathcal{A} issues a $\text{ORevealRandomness}(i, s)$, then $\pi_i^s.\text{fr}_1, \pi_i^s.\text{fr}_2 \leftarrow 0$, 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 $\text{ORevealRandomness}(i, s)$ query, then the security analysis reverts to **Case C.3**. Similarly, if \mathcal{A} issues a $\text{OCorrupt}(j)$, then $\pi_j^t.\text{fr}_2 \leftarrow 0$, and thus \mathcal{A} has no advantage in outputting $(i, s, 2, b')$. If \mathcal{A} did not issue a $\text{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.2**.
- **Case C.2:** \mathcal{A} has not issued both a $\text{OCorrupt}(j)$ and a $\text{ORevealRandomness}(i, s)$ query. Note that if \mathcal{A} issues a $\text{ORevealRandomness}(i, s)$, then $\pi_i^s.\text{fr}_1, \pi_i^s.\text{fr}_2 \leftarrow 0$, 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 $\text{ORevealRandomness}(i, s)$ query, then the security analysis reverts to **Case C.3**.
- **Case C.3:** \mathcal{A} has not issued both a $\text{ORevealRandomness}(i, s)$ and a $\text{ORevealRandomness}(j, t)$ query.

Case C.1, Game 4: In this game, we replace the computation of ck, k_2 with uniformly random and independent values \tilde{ck}, \tilde{k}_2 in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-ms-PRF-ODH challenger as described in **Case B Game 5**. Thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key \tilde{k}_2) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_2). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_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,

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

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 $\text{win} \leftarrow 1$. Thus:

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

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 $\text{OCorrupt}(j)$ and a $\text{ORevealRandomness}(i, s)$ query. In this game, we replace the computation of ck, k_1 with uniformly random and independent values \tilde{ck}, \tilde{k}_1 in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ that interacts with a sym-ms-PRF-ODH challenger as described in **Case A Game 5**. Thus:

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

Case C.2, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \text{KDF}(\tilde{ck}, g^{Ab}, 2)$. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_2 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_1) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 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,

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

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key \tilde{k}_2) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_2). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_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,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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 $\text{win} \leftarrow 1$. Thus:

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

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 $\text{ORevealRandomness}(i, s)$ and a $\text{ORevealRandomness}(j, t)$ query. In this game, we replace the computation of ck, k_0 with uniformly random and independent values \tilde{ck}, \tilde{k}_0 in the test session π_i^s and its honest partner by defining a simulator $\mathcal{B}_{\text{PRF-ODH}}$ interacting with a nn-PRF-ODH challenger identically to **Case B.2 Game 5**. Thus:

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

Case C.3, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{aB}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{aB}, 2)$. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

Case C.3, Game 6: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \text{KDF}(\tilde{ck}, g^{Ab}, 2)$. Since, by **Game 5**, \tilde{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 \tilde{ck}, \tilde{k}_2 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

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

$$\text{Adv}(\text{break}_6) \leq \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + \text{Adv}(\text{break}_7)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_1) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 3$ (i.e. when using the key \tilde{k}_2) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 9** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 3$ (i.e. when using the key \tilde{k}_2). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

$$\text{Adv}(\text{break}_9) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_{10})$$

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 7**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 10** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_i, \tilde{k}_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,

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

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

$$\begin{aligned} \text{Adv}_{\text{XX}, n_P, n_S, \mathcal{A}}^{\text{fACCE}, \text{CC.3}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(3 \cdot \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{nn-PRF-ODH}} + 4 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right) \end{aligned}$$

E.8 XK pattern

Theorem 9. *The Noise protocol XK is flexible-ACCE-secure protocol with authentication levels $\text{au} = (3, 2)$, forward-secrecy $\text{fs} = 2$, KCI resistance $\text{kc} = (3, 2)$, randomness-security $\text{r1} = (1, 3)$, eCK security $\text{eck} = \infty$, and replay resistance $\text{rp} = (2, 2)$. That is, for an adversary \mathcal{A} against the flexible ACCE security game (defined in Section 5) one can define efficient adversaries $\mathcal{B}_{\text{coll}}$ against the collision resistance of H, $\mathcal{B}_{\text{PRF-ODH}}$ against the ms-PRF-ODH, nn-PRF-ODH and sym-ms-PRF-ODH assumptions with respect to group G and KDF, $\mathcal{B}_{\text{aead}}$ against the AEAD security of AEAD, and \mathcal{B}_{prf} against the PRF security of KDF with:*

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

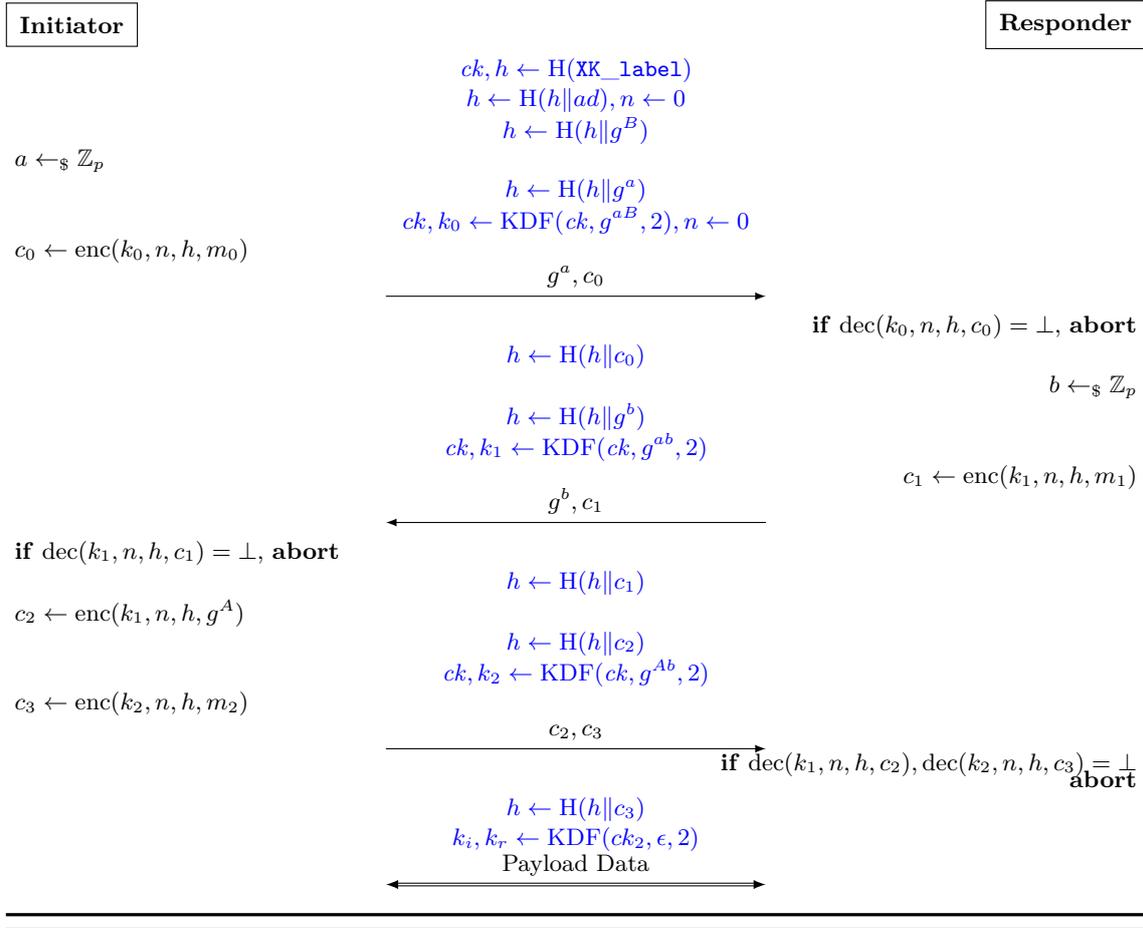


Fig. 15: Noise Pattern $\mathbf{XK} : \leftarrow s, \dots, \rightarrow e, es \leftarrow e, ee \rightarrow s, se$.

Proof.

We split the analysis into the following three cases:

- **Case A:** \mathcal{A} outputs (i, s, ζ', b') (where $\pi_i^s.\rho = \mathbf{i}$) and \mathcal{A} causes π_i^s to set $\text{win} \leftarrow 1$ when π_i^s processes the second ciphertext g^b, c_1 (i.e., when π_i^s outputs $\zeta = 2$).
- **Case B:** \mathcal{A} outputs (i, s, ζ', b') (where $\pi_i^s.\rho = \mathbf{r}$) and \mathcal{A} causes π_i^s to set $\text{win} \leftarrow 1$ when π_i^s processes the third ciphertext c_2, c_3 (i.e., when π_i^s outputs $\zeta = 3$).
- **Case B:** \mathcal{A} outputs (i, s, ζ', b') and \mathcal{A} does not cause π_i^s to set $\text{win} \leftarrow 1$ when π_i^s processes the ciphertext g^b, c_1 (if $\pi_i^s.\rho = \mathbf{i}$) or when π_i^s processes the ciphertext c_2, c_3 (if $\pi_i^s.\rho = \mathbf{r}$).

We begin by treating **Case A**.

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

$$\text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CA}} = \text{Adv}(\text{break}_0).$$

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

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

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^*, ζ', b') such that $(i^*, s^*) \neq (i, s)$. Thus:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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 $\pi_i^s.\text{pid} \neq j$. Thus:

$$\text{Adv}(\text{break}_2) = n_P (\text{break}_3).$$

Case A, Game 4: In this game, we define an abort event $\text{abort}_{\text{win}}$ if the session π_i^s sets the status $\text{win} \leftarrow 1$ 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 $\text{abort}_{\text{win}}$.

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

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, ζ', b') is output by the adversary. Similarly, by **Game 3**, we know at the beginning of the experiment the index of the intended partner $\pi_i^s.\text{pid}$ of the session π_i^s . Thus, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\text{OCorrupt}(j)$ query, as π_i^s will output $\zeta = 2$ after π_i^s processes the ciphertext g^b, c_1 , and $\text{au}^r = 2$. However, $\mathcal{B}_{\text{PRF-ODH}}$ must account for all sessions t such that party j 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: $ck, k_2 \leftarrow \text{KDF}(ck, g^{xu}, 2)$
- In sessions where the party acts as the responder: $ck, k_0 \leftarrow \text{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHu oracle provided by the ms-PRF-ODH challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. We note that $\text{au}^r = 2$, and only after processing (g^b, c_1) will π_i^s output $\zeta = 2$, and so \mathcal{A} cannot issue a $\text{OCorrupt}(j)$ 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** $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{OCorrupt}(j)$ query. In addition, since $\text{rl}^\tau = 3$ and $\text{eck} = \infty$, in **Case A** $\mathcal{B}_{\text{PRF-ODH}}$ also never has to answer a $\text{ORevealRandomness}(i, s)$ query. Thus we have:

$$\text{Adv}(break_4) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + \text{Adv}(break_6)$$

Case A, Game 6: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \text{KDF}(ck, g^{ab}, 2)$. Since, by **Game 5**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

Case A, Game 7: In this game, π_i^s will only set $\text{win} \leftarrow 1$ if \mathcal{A} is able to produce a ciphertext $g^b, c_1 := g^b, \text{AEAD.enc}(\tilde{k}_1, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{ODec}(i, s, c)$ queries directed to π_i^s when π_i^s would output $\varsigma = 2$ from \mathcal{A} (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ instead queries $\text{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable of causing $\text{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since \tilde{k}_1 is a uniformly random and independent value, this change is sound. Thus,

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

Note that the additional-data field of c_1 contains $h = \text{H}(\text{H}(\text{H}(\text{H}(\text{H}(\text{XK_label} \| ad) \| g^B) \| g^a) \| c_0) \| g^b)$. 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_i^s processing (g^b, c_1) is 0. Thus:

$$\begin{aligned} \text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CA}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S \cdot \left(\text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} \right) \end{aligned}$$

We now treat **Case B**.

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

$$\text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CB}} = \text{Adv}(break_0).$$

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

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

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:

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

Case B, 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 $\pi_i^s.\text{pid} \neq j$. Thus:

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

Case B, Game 4: In this game, we define an abort event $abort_{win}$ if the session π_i^s sets the status $\text{win} \leftarrow 1$ 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 $abort_{win}$.

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

Case B, Game 5: In this game, we replace the computation of $ck, k_2 \leftarrow \text{KDF}(ck, g^{Ab})$ with uniformly random and independent values $(\tilde{ck}, \tilde{k}_2)$ 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, c', 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, $\mathcal{B}_{\text{PRF-ODH}}$ initializes a sym-ms-PRF-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, $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{OCorrupt}(i)$ query nor a $\text{ORevealRandomness}(j, t)$ query. However, $\mathcal{B}_{\text{PRF-ODH}}$ 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: $ck, k_2 \leftarrow \text{KDF}(ck, g^{xu}, 2)$
- In sessions where the party acts as the responder: $ck, k_0 \leftarrow \text{KDF}(ck, g^{xu}, 2)$

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ 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 $\mathcal{B}_{\text{PRF-ODH}}$, as it has been generated instead by the adversary

In the second case, $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHu oracle provided by the sym-ms-PRF-ODH challenger, specifically querying $\text{ODHu}(ck, X)$, (where X is the Diffie-Hellman public keyshare such that the private key is unknown to the challenger) which will output $\text{KDF}(ck, X^u)$. However, $\mathcal{B}_{\text{PRF-ODH}}$ 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 $ck, k_0 := \text{KDF}(ck, g^{av})$, where g^a may have been contributed by \mathcal{A} . In this case, in order to compute ck, k_0 , $\mathcal{B}_{\text{PRF-ODH}}$ must instead use the ODHv oracle provided by the sym-ms-PRF-ODH challenger, specifically querying $\text{ODHv}(ck, g^a)$, which will output $\text{KDF}(ck, g^{av})$. Thus we have:

$$\text{Adv}(break_4) \leq \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}(break_6)$$

Case B, Game 6: In this game, π_i^s will only set $\text{win} \leftarrow 1$ if \mathcal{A} is able to produce a ciphertext $c_2, c_3 := c_2, \text{AEAD.enc}(\tilde{k}_2, n, h, m)$ that decrypts without error. We construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{ODec}(i, s, c)$ queries directed to π_i^s when π_i^s would output $\zeta = 3$ from \mathcal{A} (i.e. when using the key k_2). $\mathcal{B}_{\text{aead}}$ instead queries $\text{Dec}(n, h, c)$ to the AEAD challenger's oracles.

An adversary capable of causing $\text{win} \leftarrow 1$ can break the AEAD security of the AEAD scheme. Since \tilde{k}_1 is a uniformly random and independent value, this change is sound. Thus,

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

Note that the additional-data field of c_3 contains $h = \text{H}(\text{HH}(\text{H}(\text{H}(\text{H}(\text{H}(\text{XK_label} \| ad) \| g^B) \| g^a) \| c_0) \| g^b) \| c_1) \| c_2)$. 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 $\text{abort}_{\text{win}} = 1$ due to π_i^s processing (c_2, c_3) is 0. Thus:

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

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.

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

Case C, Game 1: In this game we define an abort event $\text{abort}_{\text{coll}}$ if a hash collision occurs. We do so by computing all hash values honestly and aborting if there exists two evaluations $(in, \text{H}(in))$,

$(\hat{in}, H(\hat{in}))$ such that $in \neq \hat{in}$ but $H(in) = H(\hat{in})$. The simulator $\mathcal{B}_{\text{coll}}$ interacts with a hash-collision challenger, outputting the collision if found. Thus:

$$\text{Adv}(\text{break}_0) \leq \text{Adv}_{H, \mathcal{B}_{\text{coll}}}^{\text{coll}} + \text{Adv}(\text{break}_3)$$

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:

$$\text{Adv}(\text{break}_1) = n_P n_S \cdot (\text{Adv}(\text{break}_2)).$$

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:

$$\text{Adv}(\text{break}_2) = n_P n_S (\text{break}_3).$$

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 $\text{OCorrupt}(j)$ query and an $\text{ORevealRandomness}(i, s)$ query. This allows us to prove the security of all stages ciphertexts.
- **Case C.2:** \mathcal{A} has not issued both a $\text{OCorrupt}(i)$ query and an $\text{ORevealRandomness}(j, t)$ query. Note that if \mathcal{A} issues a $\text{ORevealRandomness}(i, s)$ query at the beginning of the game, then $\pi_i^s \cdot \text{fr}_1, \pi_i^s \cdot \text{fr}_2 \leftarrow 0$, 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 $\text{ORevealRandomness}(j, t)$ query, then the security analysis reverts to **Case C.3**, since $\pi_i^s \cdot \text{fr}_2 = 1$, and we need to capture the security of the additional stage's ciphertext.
- **Case C.3:** \mathcal{A} has not issued both a $\text{ORevealRandomness}(i, s)$ and a $\text{ORevealRandomness}(j, t)$ query. Note that if \mathcal{A} issues a $\text{OCorrupt}(j)$, then $\pi_i^s \cdot \text{fr}_1 \leftarrow 0$, 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 $\text{OCorrupt}(j)$ query, then the security analysis reverts to **Case C.1** since $\pi_i^s \cdot \text{fr}_1 = \pi_i^s \cdot \text{fr}_2 = 1$, 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 $ck, k_0 \leftarrow \text{KDF}(ck, g^{aB})$ with uniformly random and independent values $(\tilde{ck}, \tilde{k}_0)$ in the test session π_i^s and its honest partner. This game proceeds identically to **Case A Game 5**. Thus we have:

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

Case C.1, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{ab}, 2)$ used to compute $ck, k_1 := \text{KDF}(\tilde{ck}, g^{ab}, 2)$. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_1 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

Case C.1, Game 6: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \text{KDF}(\tilde{ck}, g^{Ab}, 2)$. Since, by **Game 5**, \tilde{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 \tilde{ck}, \tilde{k}_2 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

Case C.1, Game 7: In this game we replace the function $\text{KDF}(\tilde{ck}, \epsilon, 2)$ used to compute $k_1, k_x := \text{KDF}(\tilde{ck}, \epsilon, 2)$. Since, by **Game 6**, \tilde{ck} is uniformly random and independent of the protocol flow, this replacement is sound. We thus replace the values k_1, k_x with uniformly random values \tilde{k}_1, \tilde{k}_x . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

$$\text{Adv}(\text{break}_6) \leq \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} + \text{Adv}(\text{break}_7)$$

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 1$ (i.e. when using the key \tilde{k}_0) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_0). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 2$ (i.e. when using the key \tilde{k}_1) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key \tilde{k}_2) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 9** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_2). $\mathcal{B}_{\text{aead}}$ 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 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,

$$\text{Adv}(\text{break}_9) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_{10})$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 7**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 10** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 k_i, k_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,

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

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 $\text{win} \leftarrow 1$. Thus:

$$\begin{aligned} \text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC.1}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(4 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{ms-PRF-ODH}} + 3 \cdot \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} \right) \end{aligned}$$

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

Case C.2, Game 4: In this game, we replace the computation of $ck, k_2 \leftarrow \text{KDF}(ck, g^{Ab})$ with uniformly random and independent values $(\tilde{ck}, \tilde{k}_2)$ in the test session π_i^s and its honest partner. This game proceeds identically to **Case B Game 5**. Thus we have:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 3$ (i.e. when using the key \tilde{k}_2) replaced in **Game 4**. In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 5** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_2). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

$$\text{Adv}(\text{break}_5) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_6)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 5**. In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_i, \tilde{k}_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,

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

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 $\text{win} \leftarrow 1$. Thus:

$$\begin{aligned} \text{Adv}_{\text{XK}, n_P, n_S, \mathcal{A}}^{\text{fACCE, CC.2}} &\leq \text{Adv}_{\text{H}, \mathcal{B}_{\text{coll}}}^{\text{coll}} + n_P^2 n_S^2 \cdot \left(2 \cdot \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} \right. \\ &\quad \left. + \text{Adv}_{\text{KDF}, G, p, \mathcal{B}_{\text{PRF-ODH}}}^{\text{sym-ms-PRF-ODH}} + \text{Adv}_{\text{KDF}, \mathcal{B}_{\text{prf}}}^{\text{prf}} \right) \end{aligned}$$

We now treat **Case C.3**.

Case C.3, Game 4: In this game, we replace the computation of $ck, k_1 \leftarrow \text{KDF}(ck, g^{ab})$ with uniformly random and independent values $(\tilde{ck}, \tilde{k}_1)$ 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, ζ', 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, $\mathcal{B}_{\text{PRF-ODH}}$ 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 and give $pk_i = g^u$ to the adversary with all other (honestly generated) public keys. Note that by the definition of this case, $\mathcal{B}_{\text{PRF-ODH}}$ never has to answer a $\text{ORevealRandomness}(j, t)$ query nor a $\text{ORevealRandomness}(j, t)$ query. However, $\mathcal{B}_{\text{PRF-ODH}}$ must account for all sessions such that $\mathcal{B}_{\text{PRF-ODH}}$ 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: $ck, k_0 \leftarrow \text{KDF}(ck, g^{uB}, 2)$, $ck, k_1 \leftarrow \text{KDF}(ck, g^{uv}, 2)$
- In sessions where the party acts as the responder: $ck, k_1 \leftarrow \text{KDF}(ck, g^{uv}, 2)$, $ck, k_0 \leftarrow \text{KDF}(ck, g^{Au}, 2)$

Dealing with $\mathcal{B}_{\text{PRF-ODH}}$'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. $\mathcal{B}_{\text{PRF-ODH}}$ can then use its own internal knowledge of the private keys to complete the computations.

Thus we have:

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

Case C.3, Game 5: In this game we replace the function $\text{KDF}(\tilde{ck}, g^{Ab}, 2)$ used to compute $ck, k_2 := \text{KDF}(\tilde{ck}, g^{Ab}, 2)$. Since, by **Game 4**, \tilde{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 \tilde{ck}, \tilde{k}_2 . Distinguishing this change implies an algorithm breaking the prf security of the key derivation function KDF, and thus:

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

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

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

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 2$ (i.e. when using the key \tilde{k}_1) replaced in **Game 4**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 6** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ queries directed to π_i^s (or π_j^t respectively) when π_i^s or π_j^t would output $\zeta = 2$ (i.e. when using the key \tilde{k}_1). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

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

In **Case C.3, Game 7**, the behavior of π_i^s is independent of the test bit $\pi_i^s.b_\zeta$, where $\zeta = 2$ and thus \mathcal{A} has no advantage in guessing this challenge bit nor in causing π_i^s to set $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when Enc and Dec would output $\zeta = 3$ (i.e. when using the key \tilde{k}_2) replaced in **Game 5**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 7** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_2). $\mathcal{B}_{\text{aead}}$ 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 \tilde{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,

$$\text{Adv}(\text{break}_7) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_8)$$

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 $\text{win} \leftarrow 1$.

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 $\text{OEnc}(i, s, m_0, m_1)$ and $\text{ODec}(j, t, c)$ queries from \mathcal{A} when `Enc` and `Dec` would output $\varsigma = 4$ (i.e. when using the keys \tilde{k}_i, \tilde{k}_r) replaced in **Game 6**). In this game, in session π_i^s we construct an algorithm $\mathcal{B}_{\text{aead}}$ that interacts with an AEAD challenger in the following way: $\mathcal{B}_{\text{aead}}$ acts exactly as in **Game 8** except responding to $\text{OEnc}(i, s, m_0, m_1)$ or $\text{ODec}(j, t, c)$ 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 \tilde{k}_i, \tilde{k}_r). $\mathcal{B}_{\text{aead}}$ 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 \tilde{k}_i, \tilde{k}_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,

$$\text{Adv}(\text{break}_8) \leq \text{Adv}_{\text{AEAD}, \mathcal{B}_{\text{aead}}}^{\text{aead}} + \text{Adv}(\text{break}_9)$$

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 $\text{win} \leftarrow 1$. Thus:

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

F Full Patterns

In this section we give a full algorithmic description of the Noise Protocol Framework Patterns.

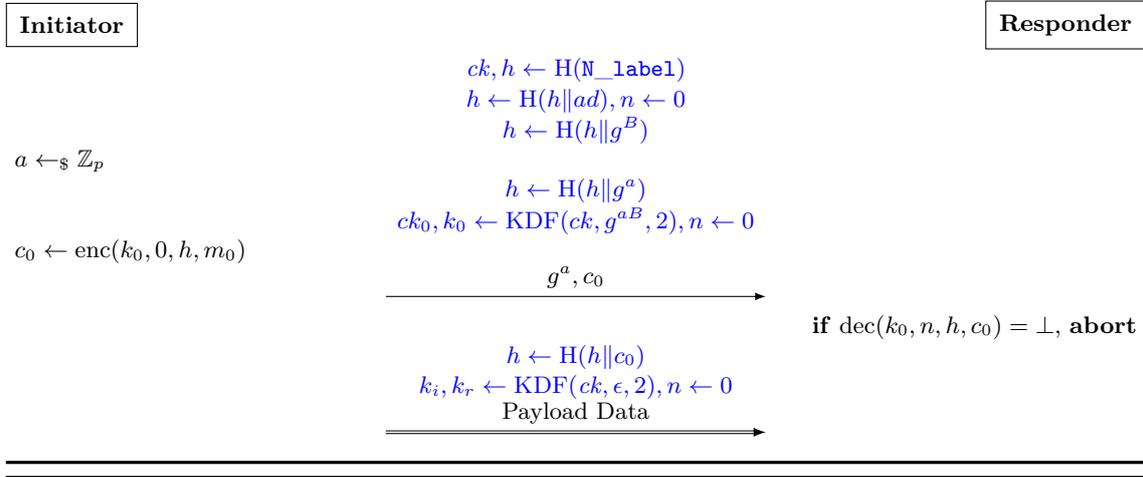


Fig. 16: Noise Pattern $\mathbf{N} : \leftarrow \mathbf{s}, \dots, \rightarrow \mathbf{e}, \mathbf{es}$.

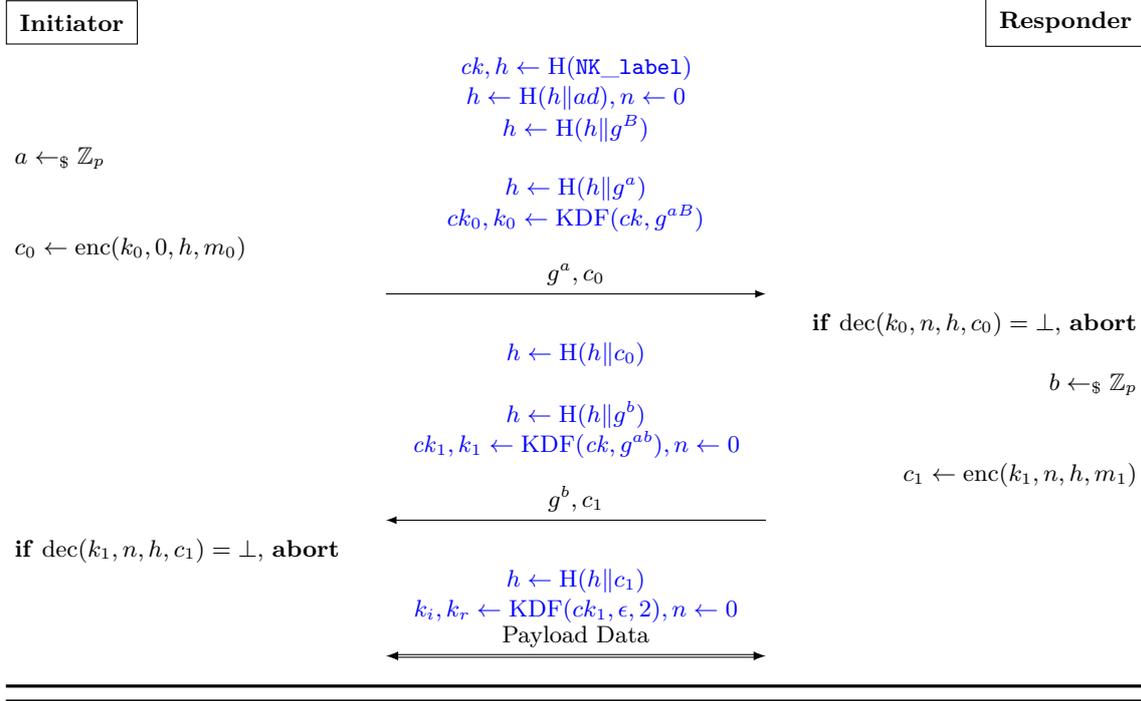


Fig. 17: Noise Pattern NK : $\leftarrow s, \dots, \rightarrow e, es$.

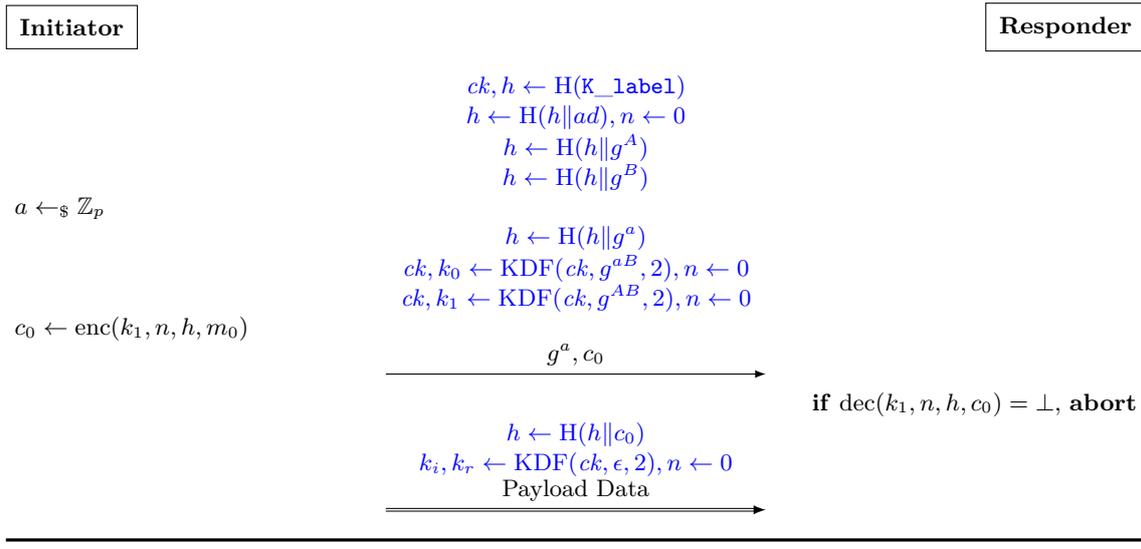


Fig. 18: Noise Pattern K : $\rightarrow s, \leftarrow s, \dots, \rightarrow e, es, ss$.

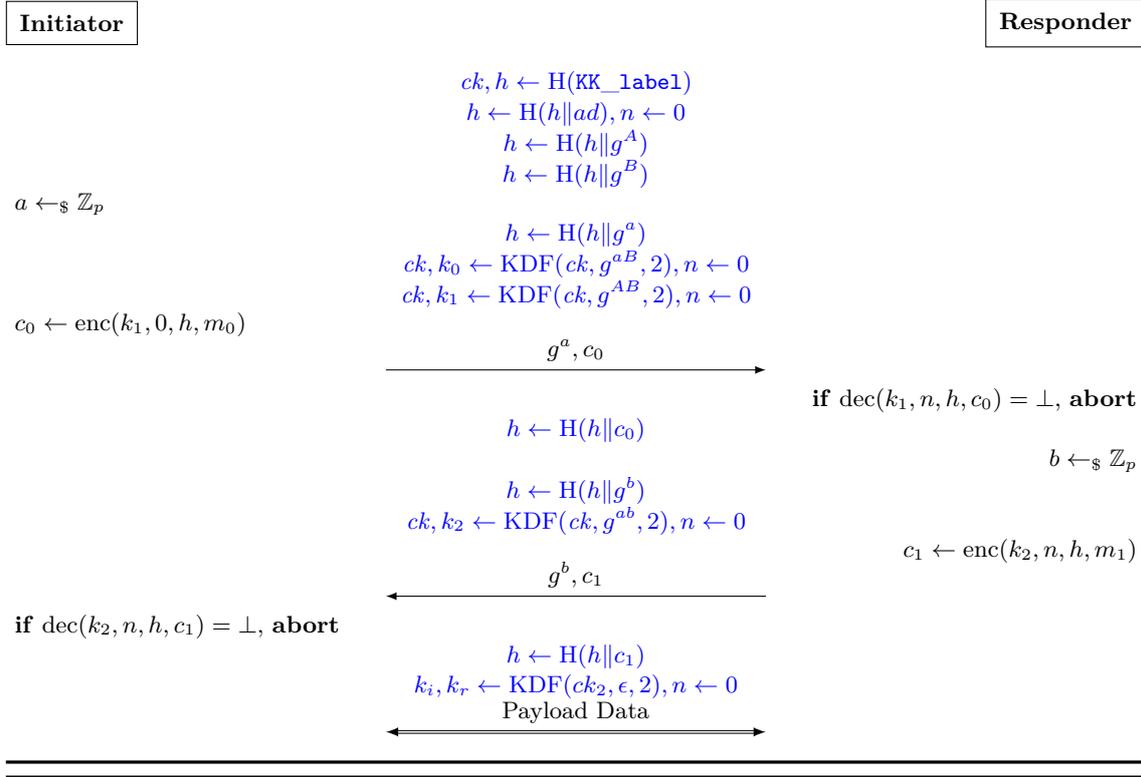


Fig. 19: Noise Pattern KK $:\rightarrow \mathbf{s}, \leftarrow \mathbf{s}, \dots, \rightarrow \mathbf{e}, \mathbf{es}, \mathbf{ss}, \leftarrow \mathbf{e}, \mathbf{ee}, \mathbf{se}$.

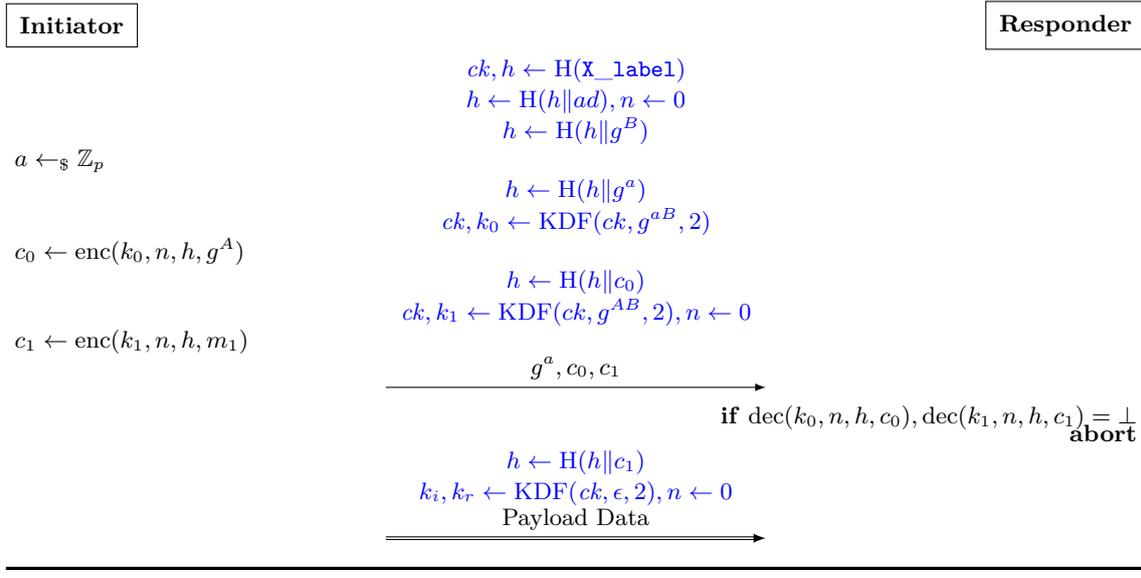


Fig. 20: Noise Pattern X $:\leftarrow \mathbf{s}, \dots, \rightarrow \mathbf{e}, \mathbf{es}, \mathbf{s}, \mathbf{ss}$.

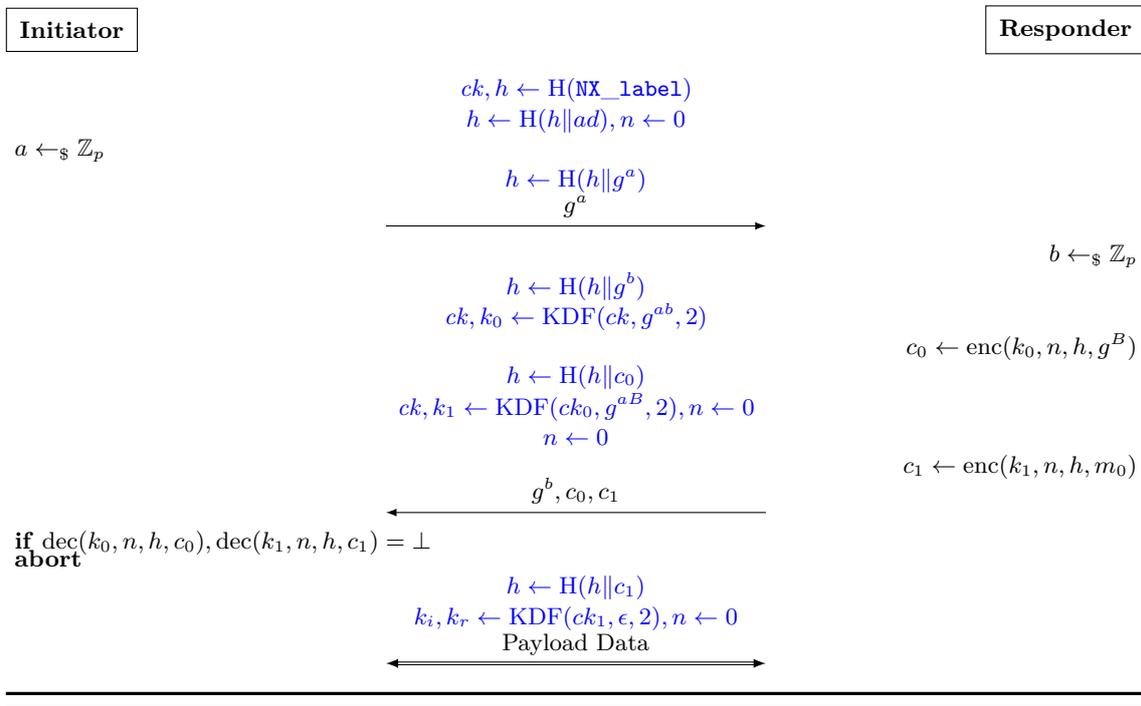


Fig. 21: Noise Pattern NX \rightarrow e, \leftarrow e, ee, s, es.

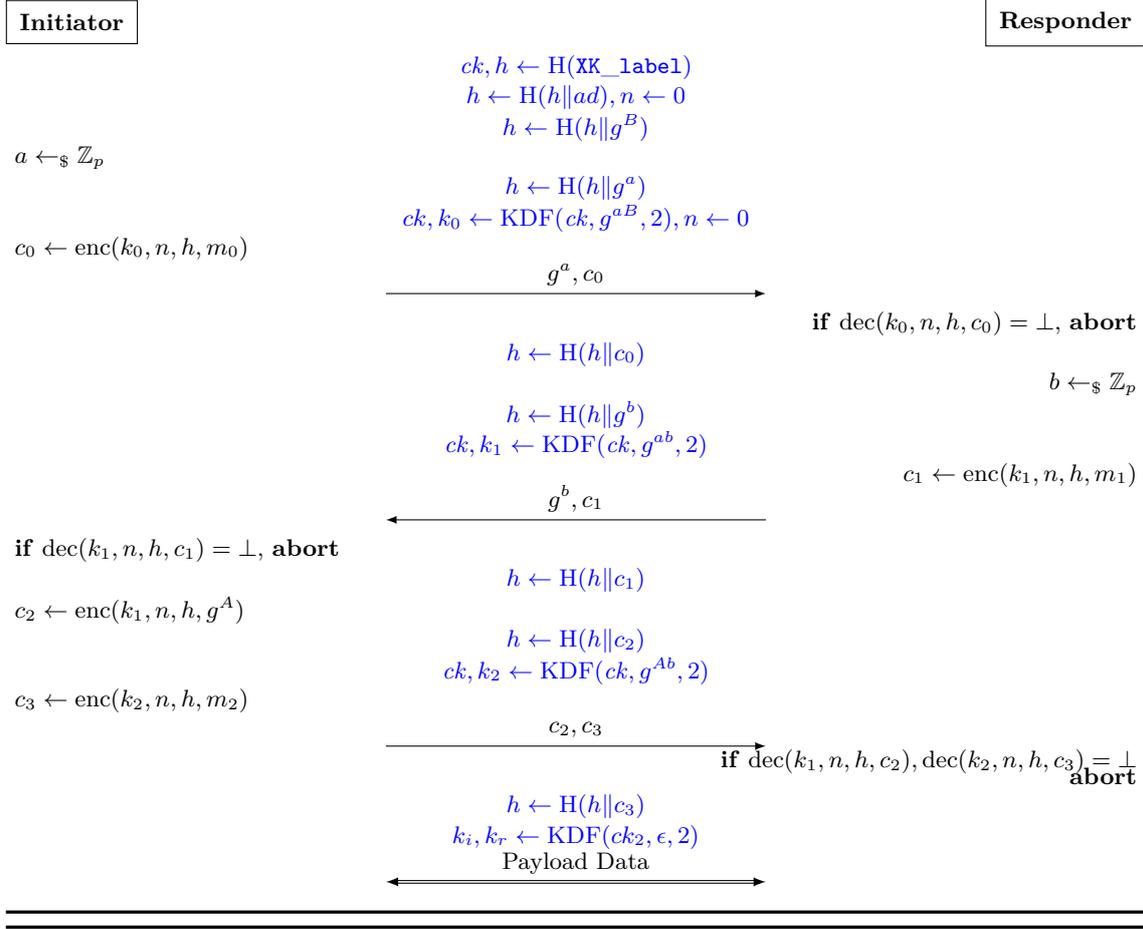


Fig. 22: Noise Pattern XK : $\leftarrow s, \dots, \rightarrow e, es \leftarrow e, ee \rightarrow s, se$.

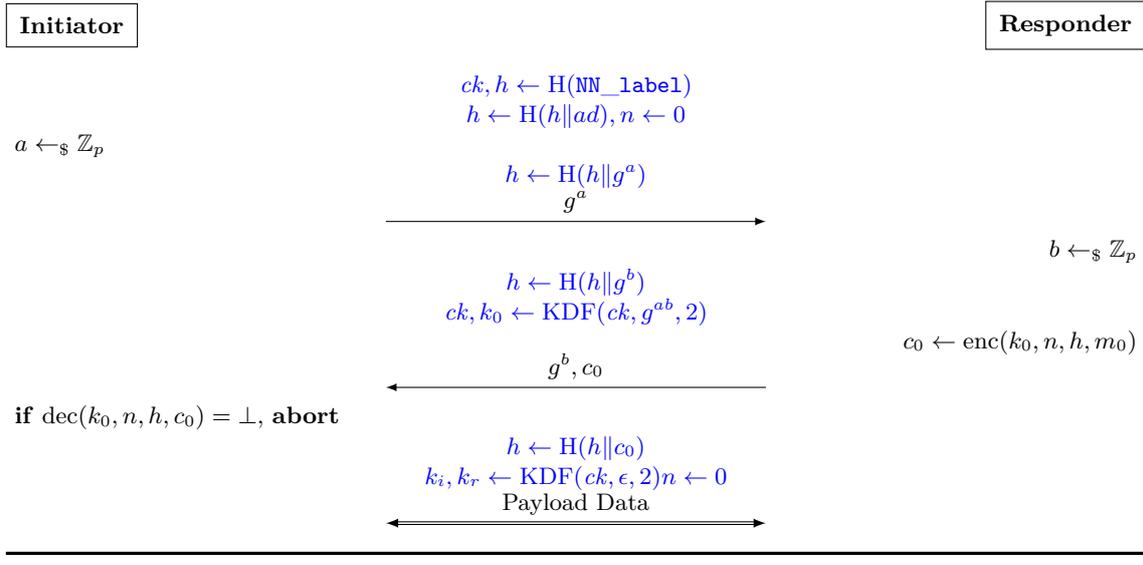


Fig. 23: Noise Pattern NN : $\rightarrow e, \leftarrow e, ee$.

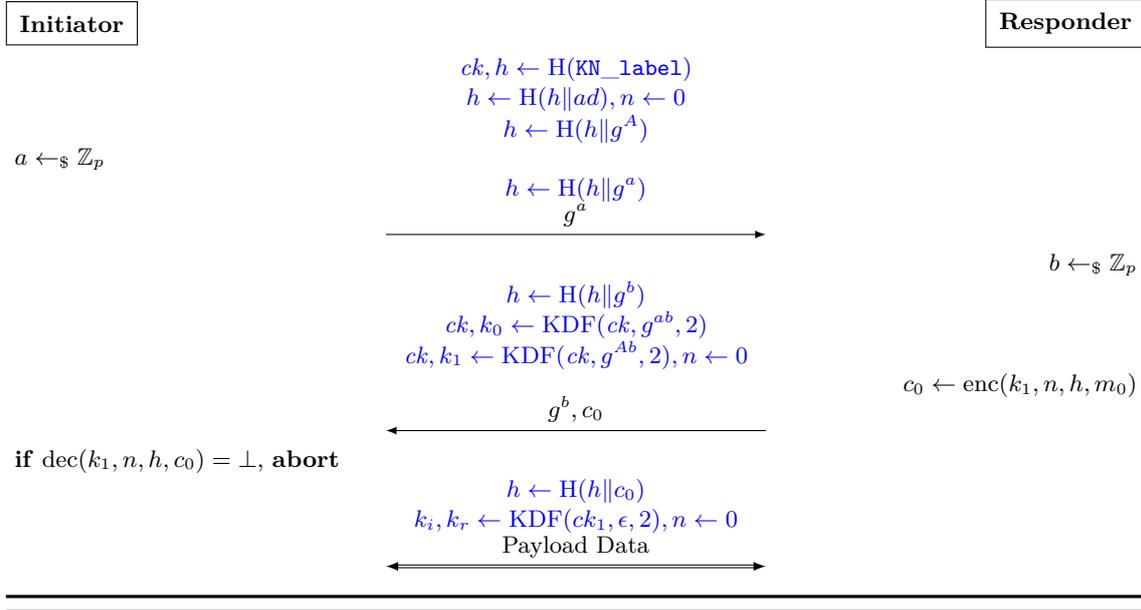


Fig. 24: Noise Pattern KN $\rightarrow s, \dots, \rightarrow e, \leftarrow e, ee, se.$

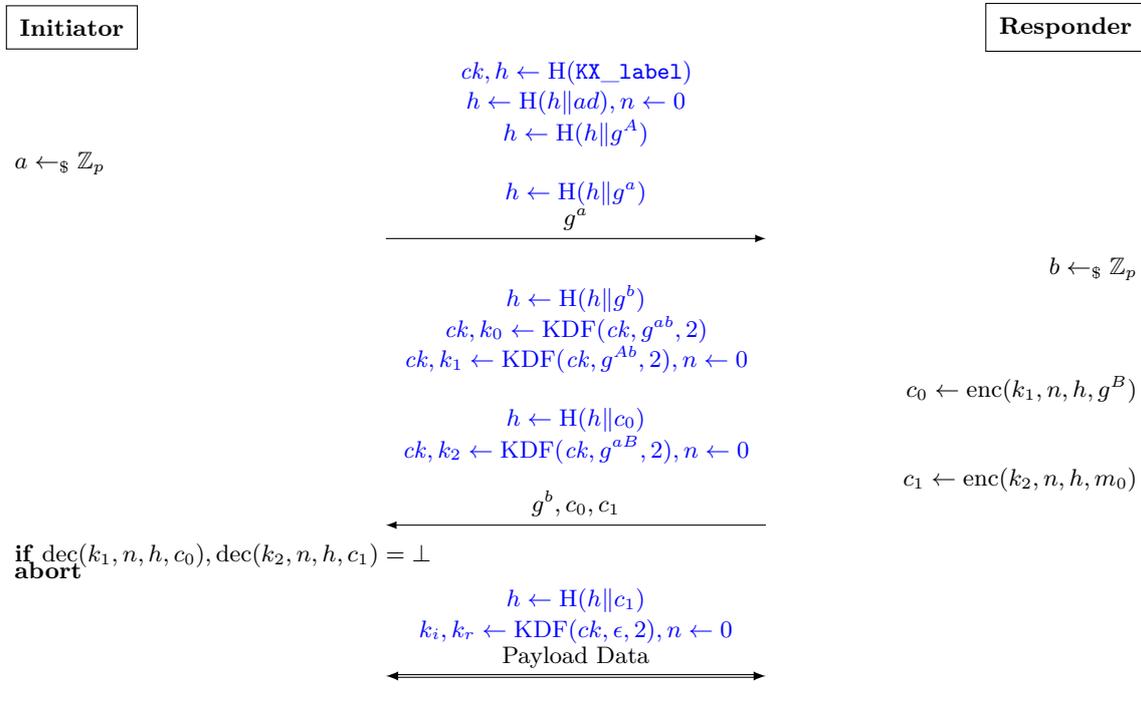


Fig. 25: Noise Pattern KX $\rightarrow s, \dots, \rightarrow e, \leftarrow e, ee, se, s, es.$

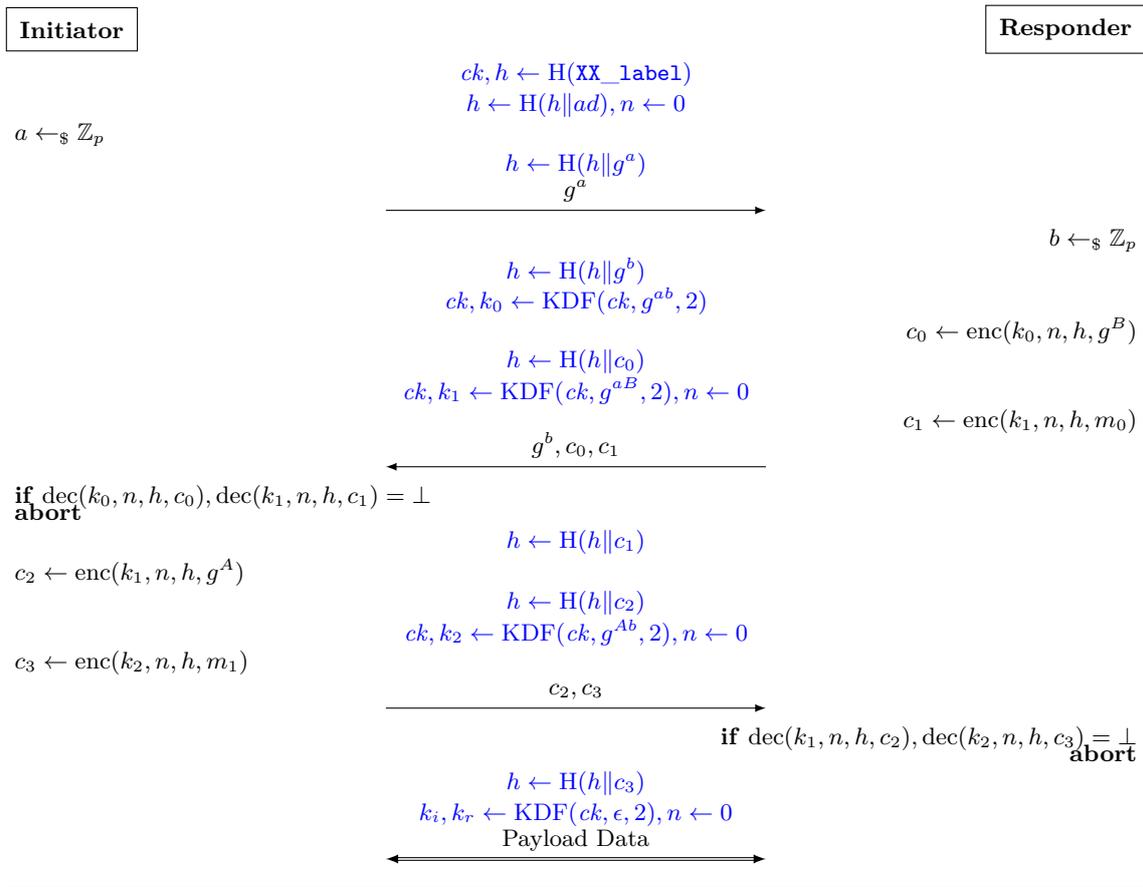


Fig. 26: Noise Pattern $\text{XX} : \rightarrow e, \leftarrow e, ee, s, es, \rightarrow s, se.$

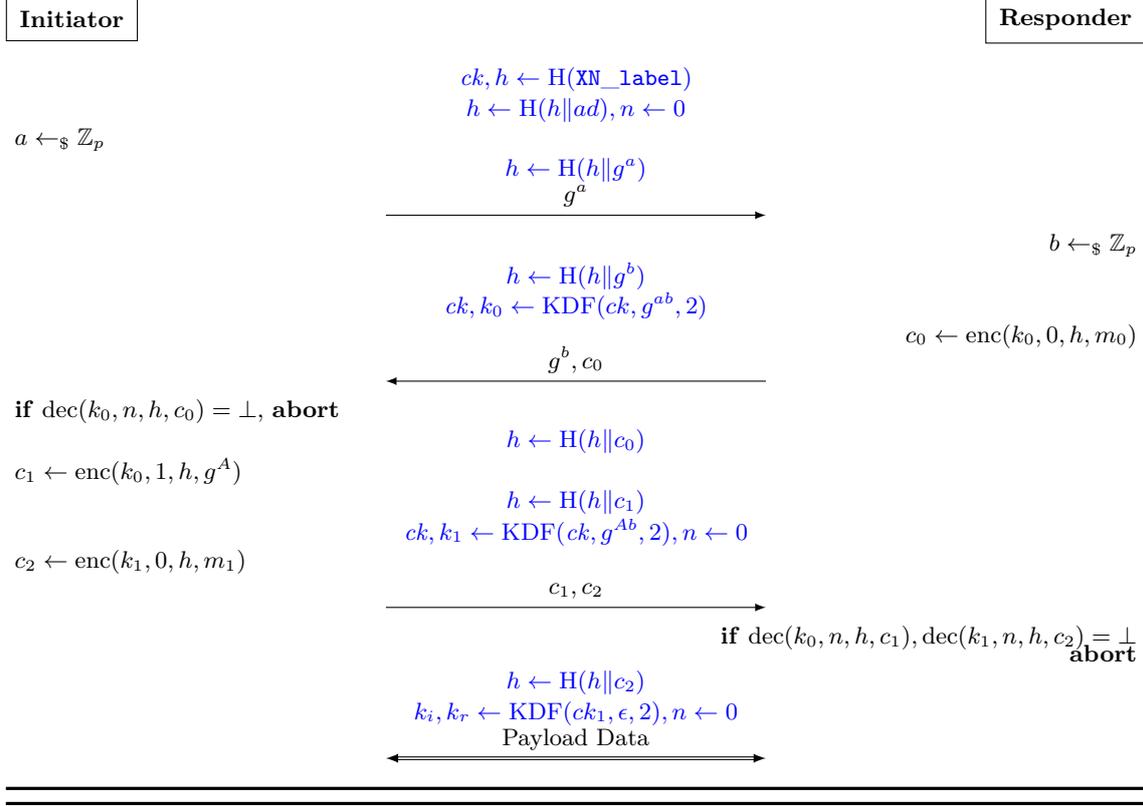


Fig. 27: Noise Pattern $\mathbf{XN} : \rightarrow \mathbf{e}, \leftarrow \mathbf{e}, \mathbf{ee}, \rightarrow \mathbf{s}, \mathbf{se}$.

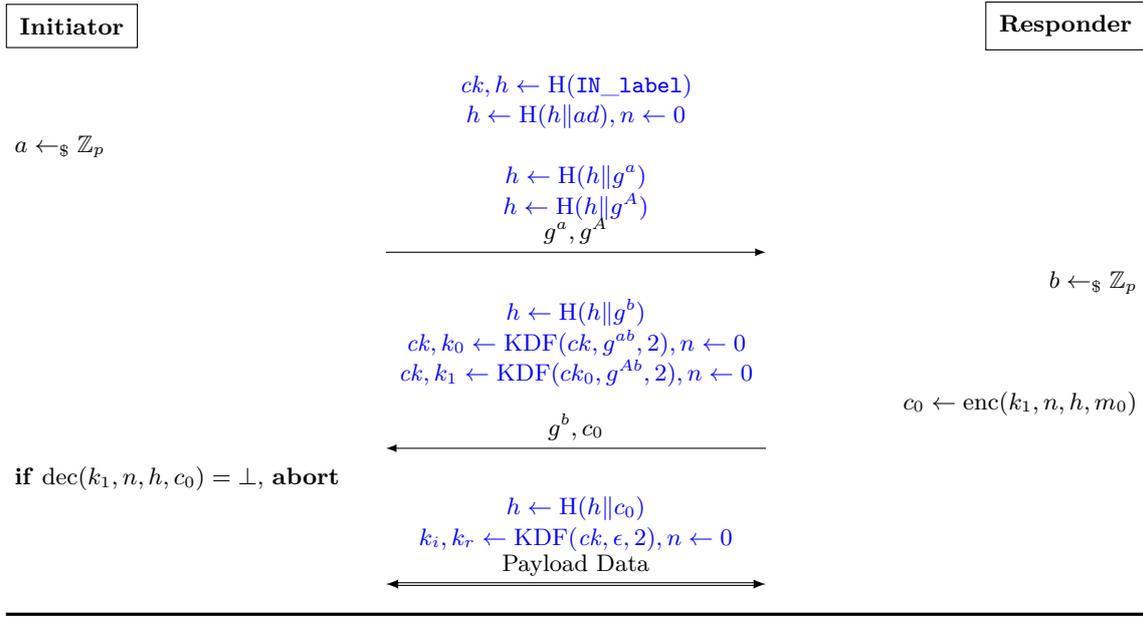


Fig. 28: Noise Pattern $\mathbf{IN} : \rightarrow \mathbf{e}, \mathbf{s}, \leftarrow \mathbf{e}, \mathbf{ee}, \mathbf{se}$.

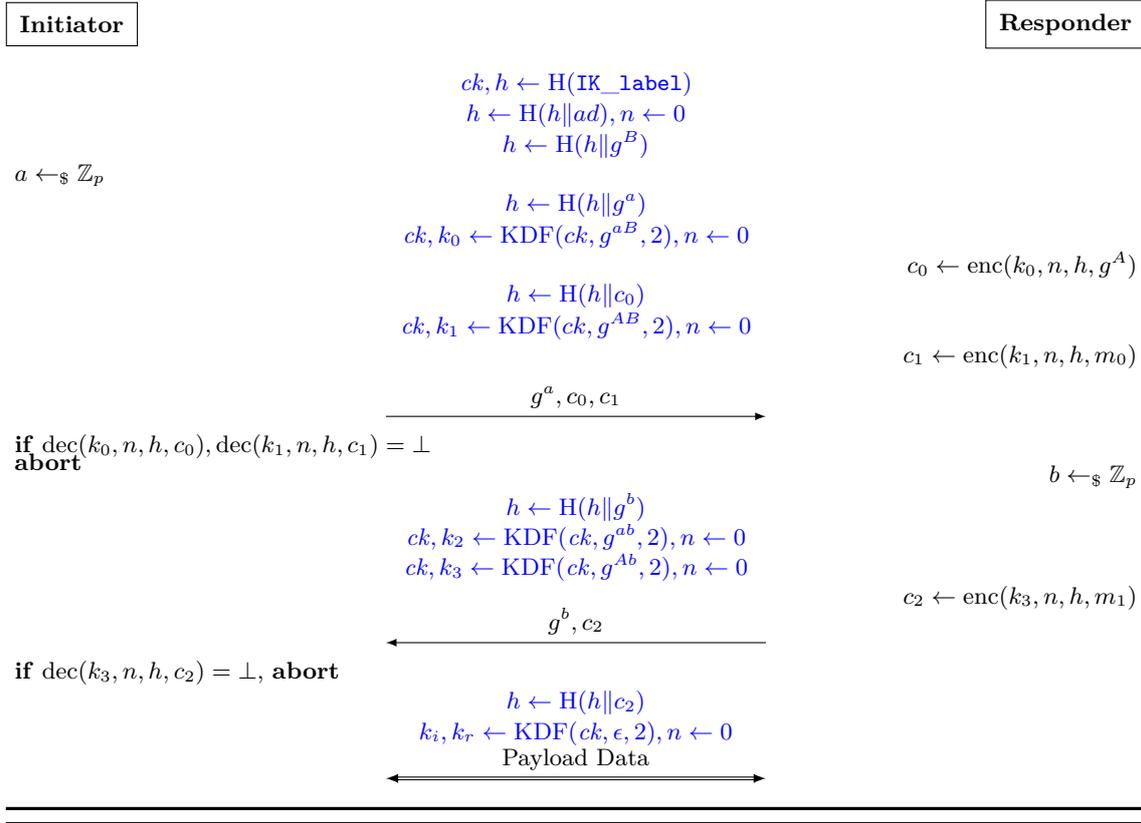


Fig. 29: Noise Pattern IK : $\leftarrow s \dots \rightarrow e, es, s, ss \dots \leftarrow e, ee, se.$

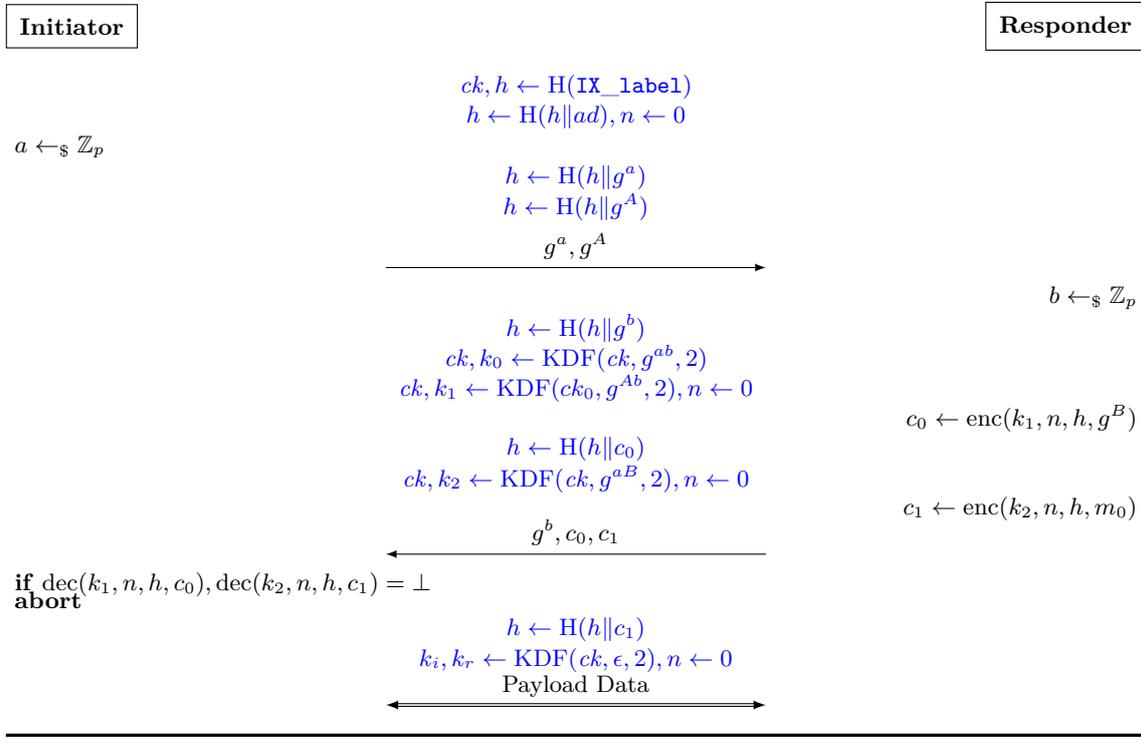


Fig. 30: Noise Pattern IX : $\rightarrow e, s, \leftarrow e, ee, se, s, es.$