A formal analysis of IKEv2's post-quantum extension

Stefan-Lukas Gazdag
genua GmbH
Kirchheim near Munich, Germany
stefan-lukas_gazdag@genua.de

Sophia Grundner-Culemann
MNM-Team, Ludwig-Maximilians-Universität München
Munich, Germany
grundner-culemann@nm.ifi.lmu.de

Tobias Guggemos
German Aerospace Centre (DLR)
Oberpfaffenhofen, Germany
tobias.guggemos@dlr.de
MNM-Team, Ludwig-Maximilians-Universität München
Munich, Germany
guggemos@nm.ifi.lmu.de

Tobias Heider
genua GmbH
Kirchheim near Munich, Germany
tobias_heider@genua.de

Daniel Loebenberger
Fraunhofer AISEC
Weiden i. d. Opf., Germany
daniel.loebenberger@aisec.fraunhofer.de

ABSTRACT

Many security protocols used for daily Internet traffic have been used for decades and standardization bodies like the *IETF* often provide extensions for legacy protocols to deal with new requirements. Even though the security aspects for extensions are carefully discussed, automated reasoning has proven to be a valuable tool to uncover security holes that would otherwise have gone unnoticed. Therefore, *Automated Theorem Proving (ATP)* is already a customary procedure for the development of some new protocols, e.g., TLS 1.3 and MLS.

IKEv2, the key exchange for the IPsec protocol suite, is expected to undergo significant changes to facilitate the integration of *Post-Quantum Cryptography*. We present the first formal security model for the IKEv2-handshake in a quantum setting together with an automated proof using the *Tamarin Prover*. Our model focuses on the core state machine, is therefore easily extendable, and aims to promote the use of *ATP* in IPsec-standardization. The security model captures gaps in the protocol, but treats the specific implementation (like fragmentation mechanisms, for example) as a black box. With IKE_INTERMEDIATE we showcase this approach on a recently proposed extension that significantly changes the protocol's state machine.

CCS CONCEPTS

• **Networks** → **Network protocol design**; • **Security and privacy** → **Security protocols**; *Public key (asymmetric) techniques.*

KEYWORDS

Formal verification, IPsec, IKEv2, ATP, quantum-resistant key exchange

ACSAC '21, December 6–10, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8579-4/21/12…$15.00
https://doi.org/10.1145/3485832.3485885

ACM Reference Format:
Stefan-Lukas Gazdag, Sophia Grundner-Culemann, Tobias Guggemos, Tobias Heider, and Daniel Loebenberger. 2021. A formal analysis of IKEv2's post-quantum extension. In *Annual Computer Security Applications Conference (ACSAC '21), December 6–10, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3485832.3485885

ACKNOWLEDGMENTS

We thank Felix Schärtl for pointing out the incompleteness of our original model, and we thank the reviewers for valuable comments and discussion, both of the paper and the artifact. Also, we gratefully acknowledge the Leibniz Super-Computing Centre (LRZ)[1] for funding this project by providing computing time.

1 INTRODUCTION

IPsec is the most popular technology for providing Virtual Private Networks (VPNs) and plays a central role for securing IP-based communication. If an IPsec connection is established correctly, the communication is then a) encrypted and thus unreadable for an attacker listening on the wire (confidential), b) transmitted such that modifications by an attacker are detected (integrity-protected), and c) guaranteed to originate from the expected communication partner (authentic). Often, further security properties apply. They are usually accomplished by the use of cryptographic schemes. Although IPsec itself uses symmetric cryptography to secure the data transfer, unacquainted parties need a key exchange protocol using asymmetric cryptography to agree on a shared secret, which is then used as the key in the symmetric scheme. Although several key agreement protocols exist, nowadays only Internet Key Exchange version 2 (IKEv2) is of practical relevance for IPsec. In this work we therefore focus on IKEv2 only.

The requirements for the whole protocol suite will change over time due to new developments in the field of networking and other new challenges. One of the motivations for future adaptions and extensions is new progress in the field of cryptography. Due to the increasing power of supercomputers, parameter sets for the cryptographic primitives employed might exceed the limits of the protocol within the coming years. Moreover, quantum computers

[1] https://www.lrz.de/

are on the rise and may pose a threat to conventional cryptography. If they reach a certain size, they will be able to break the (Elliptic Curve) Diffie-Hellman key exchange, which is essential to IKEv2. Therefore, alternative or additional quantum-safe key exchanges might be necessary. Furthermore, future governmental requisites for the use of cryptographic schemes might lead to the necessity of using multiple schemes in parallel.

In short, changes to the protocols are unavoidable to securely face future threats and demands; at the same time they are sometimes difficult to apply. The overall security level of the protocol (suite) must not decrease by introducing modifications; a proper discussion about this should be accompanied with (automated) proofs.

Contribution

We model the security of a minimal subset of IKEv2 in a quantum setting (which is when an active attacker has access to a quantum computer) and prove that in such a setting, IKEv2 is no longer secure. We also analyze a proposed extension of the protocol which allows the classic Diffie-Hellman (DH) key exchange to be supplemented by multiple (quantum-resistant) key exchanges [33, 35]. We are able to show that IKEv2 regains all desired security properties with this extension.

Our analysis allows the comparison of IKEv2 with TLS 1.3 [32] and Noise key agreement [36], which were also analyzed with Tamarin [9, 17]. It also shows that IKEv2 - at least in its minimal form - is designed so well that the extension for making it quantum-safe is both straightforward and secure.

As a by-product, we provide a Tamarin-based IKEv2-verification in the classical setting, which we use to ensure that our model agrees with the proofs in the literature regarding security in the classical case. The code is provided in Appendix A; it is open source[2] and may be used to further verify current and future protocol extensions.

A challenging aspect of the analysis is the resource-intensity: Even the correct model takes up to a half hour to compute in our environment; spotting errors is therefore laborious, and it may be difficult to spot that a model check does not terminate at all. We therefore point out that part of the contribution lies in executing the analysis.

2 BACKGROUND AND RELATED WORK

2.1 IPsec and IKEv2

IPsec is a protocol suite which describes how different security services can be used together to secure traffic in an IP-based communication system. Its architecture is standardized by the *Internet Engineering Task Force (IETF)* in RFC 4301 [24], which distinguishes between three concepts:

Security Protocol: These are the protocols used to secure the actual Internet Protocol traffic. Currently, *Authentication Header (AH)* (RFC 4302 [22]) and *Encapsulated Security Payload (ESP)* (RFC 4303 [23]) are defined, of which ESP is more commonly used. ESP achieves confidentiality, integrity and authenticity of the communication.

Security Association (SA): Security Associations (SAs) are fundamental to IPsec. Each SA defines the provided security service for a certain simplex connection between two communication peers. Importantly, it describes how the desired security goals are achieved. This includes specifying the cryptographic algorithms and keys, various session information, and the policies indicating which Security Protocol is used to secure which kind of traffic.

Key Management: The Security Protocols rely on cryptographic keys. These can be exchanged manually, but there are also protocols to establish all parameters for an SA between unacquainted communication peers, which allows for more flexibility. As stated in the introduction, we focus on Internet Key Exchange version 2 (IKEv2) as defined in RFC 7296 [20] and refer to IKEv1 only when necessary.

The IKEv2 protocol implements an authenticated DH key exchange with a minimum of two round-trips. In the first, the peers exchange public DH-values to establish a confidential channel between themselves. The second round-trip authenticates the first exchange with digital signatures or pre-shared keys. The two security protocols (ESP and AH) are designed minimalistically; the key exchange mechanism solves the most complex part of the IPsec suite. Thus, most of the currently discussed extensions of IPsec concern IKEv2.

The standardization body responsible for IPsec has never required a formal security analysis for any modifications the protocol underwent; most changes might have been regarded as too minor to prompt such effort.

However, the recently proposed drafts for integrating post-quantum key exchange to IKEv2 [33, 35] would structurally change the protocol: First, it introduces several additional round-trips in the protocol, which changes the state machine. Second, the derivation function for the session key (called `KEYMAT`, see [20, Section 2.17]) is changed to include additional key material. Structural changes like these bear a higher risk for security and should more urgently be accompanied by formal verification.

2.2 Previous analyses of IKE(v2)

IKEv1 and IKEv2 were not formally analyzed until after their respective standardization in RFC 2409 [18] and RFC 4306 [21]. The specifications explain why the protocols are believed to fulfill certain security goals, but do neither define these goals formally nor prove that they are fulfilled. A first formal analysis of IKE(v1) (using the automatic NRL Protocol Analyzer) found some security flaws in its design [29].

Additionally, IKE(v1) was subject to non-automatic formal security analysis [4]. It supplements an earlier formalization of important notions, such as *protocols* in general and *key-exchange protocols* in particular, *attackers* against such protocols as well as for *sessions* and *session-key security* [3] with a concrete example. Using this understanding, IPsec SAs can be interpreted as *sessions* for the communication partners. These definitions have since played a central role in automated verification of protocols.

IKEv2 – which obsoletes IKE(v1) – was formally modeled and verified with the Scyther tool for automatic proofs by Cremers [8]. The analysis revealed some weaknesses of IKEv2's authentication mechanism. We compare our results to his to verify the correctness

[2] An extended version of this paper can be found here [19], all code is available here: https://github.com/mnm-team/tamarin-ikev2

of our model and proofs. Apart from those attempts at formal verification of IKE and IKEv2, there exists significant work on attacker models, key exchange models, and the security properties of IKE(v2) and similar protocols. Chapter 3 refers to them in greater detail.

2.3 Formal analysis of other protocols

Formal verification has been used for other popular key exchange protocols recently. Tamarin [30, 34] was used to analyze the Noise key exchange protocol framework, which later provided the basis for a formal analysis of the Wireguard key exchange [12, 13]. It was also used frequently during the development of the TLS 1.3 standard [32]: For every revision of the proposal, the security model was adjusted and verified again [9]. Other protocols that recently received attention by formal analysis are the PKCS#11 standard [10] and the Signal protocol for private messaging with smartphones [6]. The latter motivates recent standard activities to establish a protocol for Message Layer Security (MLS), which aims to secure group communication for different smartphone messaging services [2]. A first version of the protocol is formally analyzed in [7], a procedure the IETF aims to continue during the standardization process of the new protocol.

3 PROTOCOL MODEL

IKEv2 is not only the key exchange protocol for IPsec, but also manages the IPsec SAs; this results in a multitude of extensions. Modeling all possible states, messages and error codes of the protocol would exceed the capabilities of an automatic prover, which is why we focus on a minimal subset. RFC 7815 [25] provides this in the form of Minimal-IKEv2, specifying that the agreement of keys between the peers is the minimal subset of a standard-conform implementation. Particularly, it leaves out advanced authentication modes like EAP or certificate validation. We use this subset for our model. More complex scenarios as well as adaptions and extensions of the protocol may be modeled on the basis of this later on.

Figure 1 shows the Minimal-IKEv2 state machine. **`start`** indicates that in this session, *Initiator* and *Responder* have not communicated at all yet. Every other state is reached at the end of the corresponding key agreement step:

`IKE_SA_INIT` *Initiator* The *Initiator* chooses a private ephemeral key *eI* for the DH-exchange, calculates the public ephemeral key *epI* and sends it to the *Responder* together with a `Nonce_i` in an `IKE_SA_INIT` message. The message also contains a list of proposed key exchange methods.

`IKE_SA_INIT` *Responder* The *Responder* also chooses a private ephemeral key *eR* and uses it to calculate the shared DH-key. Together with `Nonce_r` and `Nonce_i`, this shared key is used to derive the session key (called `keymat`), typically by hashing. The public ephemeral key, `Nonce_r`, and a selection of key exchange methods from the *Initiator*'s proposal comprise the *Responder*'s `IKE_SA_INIT` message, which he sends to the *Initiator*.

`IKE_AUTH` *Initiator* Upon receiving the `IKE_SA_INIT` response, the *Initiator* also calculates the shared DH key and derives the *session key* `keymat`. For authentication, the *Initiator* signs his own `IKE_SA_INIT` message with his private *static key*, thereby also proving that the `IKE_SA_INIT` message was sent by him. The `IKE_AUTH` message itself is also signed and sent to the *Responder*.

`IKE_AUTH` *Responder* The *Responder* verifies the signature with the *Initiator*'s public *static key*, and proves his own identity by signing the `IKE_AUTH` response as well as his own `IKE_SA_INIT` message with his own private *static key* and sending them to the *Initiator*.

`IKE_AUTH` Done As a last step, the *Initiator* verifies the signature of the *Responder*; this completes the exchange. Both peers now share a common SA, which can be used for communication with IKEv2 or deriving new so-called Child-SAs for other IPsec protocols.

We assume an IKEv2 session to only be valid if the final state **`IKE_AUTH Done`** is reached. We therefore prove that the states in question can be reached:

Correctness There exist sessions for both roles (*Initiator* and *Responder*) where the communication agents have established key material and those sessions can exist at the same time.[3]

Next, we define an attacker model before presenting the security properties which are to be met if such an attacker exists.

3.1 The Dolev-Yao attacker model

The Dolev-Yao attacker model [11] is very commonly used for formal analyses [13]. It allows the attacker to act with the powers of a message carrier, i.e., to eavesdrop, to hold back, resend, or modify a message, and to send fresh messages. An attacker is called "passive" if they only eavesdrop on the network. Interaction makes an attacker "active". Oracles can be introduced in the model, too, to identify weaknesses from leakage of sensitive information.

3.2 Security properties

The most important security goals of IKEv2 are informally stated to be "identity protection" and "key secrecy" [20]. The security properties of authenticated key exchange protocols have been defined and refined in several related works, among them [4], [26], [8], and [13].

We use their respective definitions of the following seven properties; changes to the cited definitions are marked by "[ ]". The properties are defined assuming that two communication partners $\mathcal{A}$ and $\mathcal{B}$ intend to establish a connection using IKEv2 with each other.

The first three properties, namely *Aliveness*, *Weak Agreement*, and *Agreement*, were first discussed by Lowe in [27] as different levels of authentication:

Aliveness *We say that a protocol guarantees to an initiator $\mathcal{A}$ aliveness of another agent $\mathcal{B}$ if, whenever $\mathcal{A}$ (acting as initiator) completes a run of the protocol, apparently with responder $\mathcal{B}$, then $\mathcal{B}$ has previously been running the protocol.*[27, Chapter 2.1] As Lowe states: "Many protocols fail to achieve even this weak form of authentication."[ibid.]

Weak Agreement *We say that a protocol guarantees to an initiator $\mathcal{A}$ weak agreement with another agent $\mathcal{B}$ if, whenever $\mathcal{A}$*

[3] Proving this statement also aids the automatic prover in proving the properties defined in Section 3.

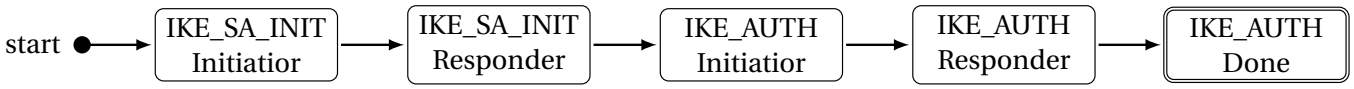

Figure 1: Minimal-IKEv2 State Machine

(acting as initiator) completes a run of the protocol, apparently with responder $\mathcal{B}$, then $\mathcal{B}$ has previously been running the protocol, apparently with $\mathcal{A}$.[27, Chapter 2.2]

Agreement (on a list of data items S) *We say that a protocol guarantees to an initiator $\mathcal{A}$ agreement with a responder $\mathcal{B}$ on a set of data items S if, whenever $\mathcal{A}$ (acting as initiator) completes a run of the protocol, apparently with responder $\mathcal{B}$, then $\mathcal{B}$ has previously been running the protocol, apparently with $\mathcal{A}$, and was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in S [..]*[27, Chapter 2.4]

Those definitions only cover the initiator's perspective, however, and leave out the possibility that Aliveness, for example, might be guaranteed to a responder as well. We therefore deviate from Lowe's approach and add the following three variants to our model (which mirror the properties for the responder):

Aliveness_R *We say that a protocol guarantees to a responder $\mathcal{B}$ aliveness of another agent $\mathcal{A}$ if, whenever $\mathcal{B}$ (acting as responder) completes a run of the protocol, apparently with initiator $\mathcal{A}$, then $\mathcal{A}$ has previously been running the protocol.*

Weak Agreement_R *We say that a protocol guarantees to a responder $\mathcal{B}$ weak agreement with another agent $\mathcal{A}$ if, whenever $\mathcal{B}$ (acting as responder) completes a run of the protocol, apparently with initiator $\mathcal{A}$, then $\mathcal{A}$ has previously been running the protocol, apparently with $\mathcal{B}$.*

Agreement_R (on a list of data items S) *We say that a protocol guarantees to a responder $\mathcal{B}$ agreement with a initiator $\mathcal{A}$ on a set of data items S if, whenever $\mathcal{B}$ (acting as responder) completes a run of the protocol, apparently with initiator $\mathcal{A}$, then $\mathcal{A}$ has previously been running the protocol, apparently with $\mathcal{B}$, and was acting as initiator in his run, and the two agents agreed on the data values corresponding to all the variables in S.*

In the remainder of the paper, we refer to the previously defined Aliveness, Agreement, and Weak Agreement as "Aliveness_I", "Agreement_I", and "Weak Agreement_I" when the properties are analyzed from the initiator's perspective, and only use the non-indexed terms like "Aliveness" when the perspective does not matter.

Lowe names "*full agreement*" (which means "agreement on all the atomic data items used in the protocol run" [27]) as "the most useful definition of authentication"[ibid.]. In our model, we consider the session key and a session identifier to be the only items on the list S; we thus define authentication as *Agreement* on the session key and session identifier.

To achieve *full agreement* in a given session, the set of values S that $\mathcal{A}$ and $\mathcal{B}$ agree on must define a unique run of the protocol between the two parties, as it states in Lowe's definition of Agreement: "[..] and each such run of $\mathcal{A}$ corresponds to a unique run of $\mathcal{B}$" [27]. This avoids an attack where one of the agents is tricked into believing that they have two (identical) sessions with their peer while in reality their peer is only running one corresponding instance. To comply with this understanding of authentication our model includes the property *session uniqueness*:

Session Uniqueness *Different sessions [with the same communication partners] will always have different, unique session keys [..].*[13, Section 3.3]

An idea closely related to *Session Uniqueness* is *Consistency*:

Consistency *If two honest parties establish a common session key then both need to have a consistent view of who the peers to the session are. Namely,if a party $\mathcal{A}$ establishes a key $\mathcal{K}$ and believes the peer to the exchange to be $\mathcal{B}$, then if $\mathcal{B}$ establishes the session key $\mathcal{K}$, it needs to believe that the peer to the exchange is $\mathcal{A}$; and vice-versa.*[26, Section 2.1]

The most obvious goal of IKEv2 is keeping the session key (with which the peers' future communication shall be protected) safe from an attacker. To formalize this we follow [8] and define *Secrecy* in general:

Secrecy (of a term t) *The term t, e.g., a computed session key, will not become known to the adversary.*[8, Chapter 3]

Key secrecy is therefore achieved if, and only if, *Secrecy of the session key* is guaranteed. The last property, *Identity Protection*, differs from the other requirements as it is only an additional security guarantee. As noted in [26], the security of the IKEv2 protocol does not directly depend on *identity hiding*. The protocol was designed to provide *Identity Protection* for the responder against an active attacker by only including the identity value in a reply to an authenticated initiator.

Identity Protection *[The goal is to] protect both identities from passive attacks and [..] protect the identity of one of the peers from disclosure against an active attacker.*[26, Section 2.2] This coincides with the definition of "identity concealment" in [4]. Both sources note that it is impossible to provide identity protection against an active attacker for the first-authenticating agent (i.e., the initiator in IKEv2): Disclosure of one's identity is inevitable for authentication and an active attacker in the role of the responder can legitimately request the initiator's identity without authenticating themself. For the responder, "Identity Protection" amounts to "secrecy of its identity".

4 AUTOMATED PROOF OF IKEV2

The authenticated key exchange as the core feature of IKEv2 is a clearly defined state machine (see Fig. 1). It therefore naturally allows automated proofs in a *multiset rewriting* system, which is often used in security protocol analyses (e.g., see [5, 16]). Thereby, the protocol's state is a multiset of facts and the protocol itself is modeled as a set of rewriting rules. Tamarin is a formal verification tool which supports automated proving of protocols by using its

own modeling language. The language is based on (labeled) multiset rewriting rules [34] and therefore naively allows to express IKEv2's state transitions as such rules. By default, the adversary is modeled as a Dolev-Yao attacker, against which we defined IKEv2's security model.

4.1 The Tamarin Prover

Tamarin's rule set and built-in cryptographic functions allow straightforward modeling of security protocols. The tool contains predefined components which model the Diffie-Hellman key exchange, hash functions, symmetric and asymmetric encryption, and public key signatures. The automatic solver uses constraints solving and multiset rewriting techniques to perform a comprehensive symbolic search for execution paths that satisfy the provided constraints and rules. Found traces and proofs can be visualized in auto-generated graphs via Tamarin's web interface.

4.1.1 Protocol Modeling. A protocol is modeled in the form of an equational theory, the corresponding multiset rewriting system, and guarded formulas which can be checked for validity or satisfiability for the traces in the system. For a meaningful analysis, it is important that the model be as close to the actual protocol definition as possible and models all security-relevant information and events correctly. However, the resource requirements of the Tamarin verification increase with the complexity of the model, thus a trade-off has to be found between the preciseness of the model and abstraction for the sake of successful analysis.

For this reason we keep the complexity at a minimum by settling on a basic subset of the IKEv2 protocol as presented in Section 3.

4.1.2 State Machine. Tamarin rules define allowed state transitions in the protocols, as depicted in Fig. 1. The state at any point in time is defined as the combination of internal state of initiator I and responder R, the last message sent over the public channel, and any persistent public knowledge such as public keys. Which state transitions are allowed in the model is defined by so-called "rules".

The following paragraph illustrates Tamarin's description language based on a simple example:

```
rule Example:
  [Fr(~sk), !Message(m)] //Premise
  -->
  [Out(senc(m, ~sk))] //Conclusion
```

The rule "Example" models the encryption of a message m with the symmetric secret key sk. Each rule consists of a premise and conclusion, marked with `[ .. ]`, that are separated by an arrow `-->`. The global state – a *multiset* – is modeled with so-called facts; in the example these are `Fr(~sk)` and `!Message(m)`. The example shows two special facts, `Fr()` denotes a *fresh* fact, that is created for the rule and can be consumed only within the rule as sk (`~sk` denotes a new name for the fact). Persistent facts (denoted with `!`) can be consumed arbitrarily often.

Transmitted is the result of `senc(m, ~sk)`, where `senc(·,·)` is the built-in symmetric encryption functions; hence `senc(m, ~sk)` models the result of symmetrically encrypting the variable `m` with the newly generated secret key `~sk`. (The corresponding decryption function is `sdec(·,·)`.)

Transmission of the resulting ciphertext is modeled by the built-in fact `Out()`, denoting that the message is observable by the Dolev-Yao attacker and the recipient.

4.1.3 Attacker Model. The Tamarin prover by design supports the Dolev-Yao attacker model [34]. The attacker is allowed to receive any messages denoted by the built-in fact `Out` used in the *rules* to model the communication. For proving the security properties under certain assumptions, the knowledge gained by the attacker is presented as the built-in fact `K(m)`. We use the latter to model *Identity Protection* and Key secrecy. In that regard, our implementation follows previous usages of Tamarin for other protocols, described in [13] and [9].

4.1.4 Lemmata. Statements about the protocol that shall be proven, like the security properties of IKEv2, are defined with the keyword `lemma` in the form of trace properties. They can be automatically proven or disproven by Tamarin's automatic solver. For this purpose, the properties have to be defined as action facts in the corresponding rules.

An example of such lemma could be:

```
lemma plain_secrecy:
not (
Ex plain #i #j .
Received(plain) @ #i
& K(plain) @ #j
)
```

The special fact `K(plain)` denotes `plain` is known to the attacker, # marks temporal names. The lemma reads as:

> *There does not exist a value* `plain`*, and points in time* `#i` *and* `#j` *with* `plain` *being received by the peer in* `#i` *and the adversary knowing the plain text in* `#j`.

4.2 Execution Environment

We implement IKEv2's state machine as Tamarin rules and the security properties described in Section 3 as Tamarin lemmata.

The analysis of the IKEv2 model was enabled by the use of the high performance compute cloud hosted at the Leibniz Supercomputing Centre (LRZ) in Garching near Munich, Germany. The analysis ran on a virtual Ubuntu Linux machine leveraging 40 CPU cores and 180 GB of RAM. Even though automated analysis has become more efficient in recent years, this set-up proved to be invaluable for some of the computationally harder lemmata. In the following we describe our findings and results of the formal verification in the classical setting using the Tamarin prover.

4.3 Verifying the Correctness of our Model

To show the correctness of the IKEv2 model, two lemmata are defined which prove that it is possible to successfully complete IKEv2 handshakes between two peers. This property is necessary for the analysis because if it was not fulfilled, all further proofs would falsely succeed simply because their conditions could never be reached by the automated solver.

exists_session *This lemma verifies that the two peers can complete a session that results in an identical shared session key for both peers.*

Table 1: Implications of key compromise: Each entry indicates whether the security property of the corresponding row is achieved (✔) or not (✗) if the key of the corresponding column is compromised.

	none	I eph.	R eph.	I static	R static
Aliveness_I	✔	✔	✔	✔	✗
Aliveness_R	✔	✔	✔	✔	✗
Weak Agreement_I	✔	✗	✗	✔	✗
Weak Agreement_R	✗	✗	✗	✗	✗
Agreement_I	✔	✗	✗	✔	✗
Agreement_R	✗	✗	✗	✗	✗
Consistency	✔	✔	✔	✔	✗
Key secrecy	✔	✗	✗	✔	✗
Identity Protection	✔	✗	✗	✗	✔

exists_two_sessions *This lemma shows that two peers can perform at least two successive handshakes with each other. This property is crucial as some vulnerabilities might require to leak information in one session and reuse it in a second.*

The IKEv2 model succeeds at fulfilling both lemmata.

4.4 Verifying IKEv2's Security Properties

We verify the security properties established in Section 3 in consecutive steps. First, we show which ones IKEv2 achieves in the standard model of the Delov-Yao attacker. Next, we allow the different cryptographic keys established between Initiator I and responder R during the key exchange to be leaked. This allows a fine-grained analysis of the security properties achieved by IKEv2. Additionally, we can validate our findings – and thereby our Tamarin model – by comparing it to the work of Cremers [8], which uses the same approach but another automatic prover, and to [37], which compares the results from several automated provers (including Tamarin).

Table 1 summarizes our findings which correspond with the ones in [8]. The first column shows that if no keys are compromised, IKEv2 fulfills all presented security properties for the initiator, but cannot guarantee Weak Agreement or Agreement to the responder. Next, we let the attacker compromise the initiator's ephemeral key, i.e., her private share for the DH-exchange. We model this as a Tamarin rule `reveal_dh` leading to the fact `RevDH`, which is then used in the corresponding lemmata of the security properties:

```
rule reveal_dh:
  [ !DHtoReveal($I,k) ]
  --[RevDH($I)]->
  [ Out(k) ]
```

In that case, the proof shows violation of the property Weak Agreement for the initiator, too. This implies that Agreement is also not achievable. Additionally, the knowledge of the Initiator's private *ephemeral key* allows the attacker to obtain the shared secret and, hence, violates Key secrecy. This allows the attacker to silently impersonate [1] the initiator; the responder's identity becomes unprotected and Identity Protection is violated. The same argumentation and violations apply when the responder's ephemeral keys are leaked.

In a third and last step, we let the attacker compromise the peer's static keys – which are the keys used for authentication. We modeled this with another Tamarin rule `reveal_static`, leading to the fact `RevSk`, and use it in the corresponding lemmata:

```
rule reveal_static:
  [ !PrivKey($I,sk) ]
  --[RevSk($I)]->
  [ Out(sk) ]
```

In the case of the Initiator's static keys being leaked, all previously achieved security properties except Identity Protection stay intact. Identity Protection is not achieved because this leakage allows an attacker to perform an impersonation attack [1].

Leakage of the responder's static key shows the opposite findings, as none of the security properties except Identity Protection are achievable in that case. As in the previous case, such leakage allows an impersonation attack, but this time it is reversed. The attacker can take over the role of the *Responder*, undetectable by the *Initiator*. Hence, Aliveness is not achievable, which implies that Weak Agreement and Agreement are violated as well. The impersonation allows the attacker to act as a man-in-the-middle. Thus, the peers will use different session keys, which violates Consistency and Key secrecy.

Notably, IKEv2's Aliveness and Consistency are violated if, and only if, the responder's static key is leaked. Even with the combined leakage of both ephemeral and the initiator's static key, these two properties are achieved.

4.5 Results

Our analysis confirms the results of former formal verification and aided analysis of IKEv2, such as [8], [37], and [31]. For each of the security properties, the Tamarin verification comes to the same results as previous publications. Hence, we show that the IKEv2 protocol satisfies all desired security properties unless one or more of the secret keys have been compromised. These identical results at the same time confirm the validity of the presented IKEv2 model. The properties of Aliveness and Consistency are only violated in a single attacker setting and, thus, are especially robust.

4.6 Ambiguity in Authentication properties

In A.8, we provide a variation of the model which takes into account that "running the protocol" may be understood as "completing it" (i.e., sending the last own message of the protocol) rather than only "executing any steps of the protocol". In that case, Aliveness can only be achieved for the initiator (Aliveness_I) but not the responder (Aliveness_R). By this interpretation, the authentication of the initiator is therefore quite weak.

5 AUTOMATED PROOFS IN A QUANTUM SETTING

With the basic model of IKEv2 for the Tamarin Prover established for the classical case in the previous section, we can now extend

the proof to model deviations from the core protocol and move to a quantum-setting.

Fig. 2 shows the state machine presented before (see Fig. 1) in vertical form and with some possible extensions to the core protocol: On the right, **COOKIE** [20] (marked orange), used in the first phase of the key exchange to re-use parameters from previously established SAs, and **EAP** [14] (marked red), a special authentication protocol which changes the order in which the peers authenticate themselves.

On the left (marked blue), the IKE_INTERMEDIATE extension [33] is illustrated. It is one of the most recently proposed extensions and was drafted with the need for hybrid (quantum-safe) key exchanges in mind [ibid.]. Among other uses, it allows the IPsec-key exchange to support multiple key exchanges (as specified in draft [35]) and thus enables the use of strong post-quantum cryptography (such as the McEliece-cryptosystem [28]) in the context of IKEv2.

While we assume that formally proving the security for any extension would be valuable feedback to the community, most of them have been established for years without any noteworthy security breaches. We therefore focus on the two extensions mentioned above, which are still in draft status and do not enjoy quite the same level of confidence yet.

5.1 Extending the Tamarin Model

The IKE_INTERMEDIATE exchange takes place after IKE_SA_INIT but before IKE_AUTH and consists of one or more round-trips; for simplicity we model an exchange with one round-trip only. Extending the model established in Section 4 accordingly requires two new Tamarin rules that implement the respective new states for the *Initiator* and *Responder*. The rules IKE_AUTH_I and IKE_AUTH_R also change, as their input condition is now the successful completion of the new exchange; a session in this model is therefore only complete when IKE_INTERMEDIATE is used.

The attacker in our basic model is able to reveal classic DH-keys; it is with respect to her that the security properties in Section 3 were proven to hold. To account for quantum-resistant key exchanges, too, we extend the attacker model by the following rule, which allows the revelation of quantum-resistant DH-keys:

```
1 rule reveal_dhq:
2 [ !DHQtoReveal($I, k) ]
3 --[RevDHQ($I)]->
4 [ Out(k) ]
```

The new fact !DHQtoReveal(X, k) is used in the IKE_INTERMEDIATE_X-rules to indicate that peer X has created a quantum-safe DH-key.

5.2 Verifying PQ-IKEv2's Security Properties

The additional key exchange should protect the peers' shared key against even a quantum-computer-based attacker [35], i.e., the enhanced IKEv2-version should provide Key secrecy against this attacker. Ideally, it should feature all the security properties that the original IKEv2 provides against a classical attacker.

To prove this, we use the same methodology as presented in Section 4.4. The Tamarin lemmata are adjusted to account for the new attacker model; this affects all properties which are only achieved if the DH-values are not revealed, namely: a) Weak Agreement, b) Agreement, c) Key secrecy and d) Identity Protection. In all of the corresponding lemmata, it is necessary to assume that the DH-keys are not revealed to prove that the property holds. A quantum-computer-based attacker can reveal the classical DH-keys by definition. For all properties in question, it is instead necessary to assume that the quantum-resistant DH-keys are not revealed to prove that they still hold.

In the lemma for Weak Agreement_I, for example, the fact RevDH(X) needs to be replaced by RevDHQ(X) as follows (see Lines 5 and 6):

```
1 lemma weak_agreement_i[use_induction]:
2   "All spi I R keymat #i .
3   Completed(spi, I, 'initiator', R, keymat)
        @ #i
4   & not (Ex #k . RevSk(R) @ k)
5   & not (Ex #k . RevDHQ(I) @ k) //<-
        Post-Quantum
6   & not (Ex #k . RevDHQ(R) @ k) //<-
        Post-Quantum
7   ==> (Ex spi2 role nonce keymat2 #j .
        Agreed(spi2, R, role, I, nonce,
        keymat2) @ #j
8   & #j < #i)"
```

The lemmata for the other properties must be adjusted in the same manner. Section A.7 of the appendix offers a complete overview of the adjusted code segments.

5.3 Results

In the model for IKE_INTERMEDIATE with quantum-safe keys and a quantum-computer-based attacker, the security properties specified in Section 3 can be proven to hold under the equivalent conditions; i.e., if, and only if, the leakage of a classical DH-key violates a security property in the classical model, then the leakage of the quantum-safe DH-key violates the same property in the quantum-safe model. This also shows that a one-round-trip IKE_INTERMEDIATE-exchange does not allow any additional attacks with respect to the proposed security model.

6 CONCLUSION AND FUTURE WORK

Using the automated proof system Tamarin, we provide the first formal security proof of the Minimal-IKEv2 protocol in a Dolev-Yao style attacker model that includes access to a quantum computer and the first formal analysis of IKE_INTERMEDIATE used for post-quantum key exchange in IKEv2.

In our analysis we cover properties such as the secrecy of the computed session key and identity protection for the communicating parties. The presented results for the classical setting are consistent with existing work that uses other automated or manual proof systems, giving our work validity.

Our approach allows modeling extensions of the IKEv2 protocol in a relatively simple manner. With only few changes, therefore, we move our attacker model to a quantum setting and show that

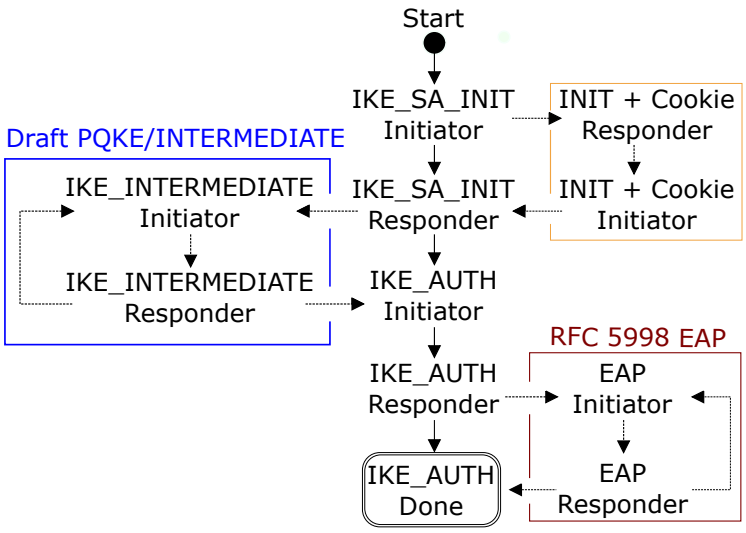

Figure 2: A comprehensive graph of the IKEv2 handshake's state machine including optional extensions to the model exemplarily [14, 15, 20, 33, 35].

in this case, the IKEv2 protocol extension `IKE_INTERMEDIATE` preserves the security properties of the original when it is used for an additional, one-round post-quantum key exchange.

To support the future standardization work around IPsec and IKEv2, we plan to verify the security properties of other extensions as well. In our model, we explicitly assume that there is exactly one intermediate key exchange. In future work one might want to extend this to the general case of (potentially) multiple `IKE_INTERMEDIATE` exchanges. Authentication with certificates[14] (which is quite resource-intensive) and other complex authentication mechanisms may also be analyzed, and in the long run, quantum-safe authentication should be, too.

REFERENCES

[1] Carlisle Adams. 2005. *Impersonation Attack*. Springer US, Boston, MA, 286–286. https://doi.org/10.1007/0-387-23483-7_196

[2] Richard Barnes, Benjamin Beurdouche, Jon Millican, Emad Omara, Katriel Cohn-Gordon, and Raphael Robert. 2019. *The Messaging Layer Security (MLS) Protocol*. Internet-Draft draft-ietf-mls-protocol-08. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-08 Work in Progress.

[3] Ran Canetti and Hugo Krawczyk. 2001. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 453–474.

[4] Ran Canetti and Hugo Krawczyk. 2002. Security analysis of IKE's signature-based key-exchange protocol. In *Annual International Cryptology Conference*. Springer, 143–161.

[5] Iliano Cervesato, Nancy Durgin, Patrick Lincoln, J Mitchell, and Andre Scedrov. 2002. A comparison between strand spaces and multiset rewriting for security protocol analysis. In *International Symposium on Software Security*. Springer, 356–383.

[6] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2017. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 451–466.

[7] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2017. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. Cryptology ePrint Archive, Report 2017/666. https://eprint.iacr.org/2017/666.

[8] Cas Cremers. 2011. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In *European Symposium on Research in Computer Security*. Springer, 315–334.

[9] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. 2017. A comprehensive symbolic analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1773–1788.

[10] Alexander Dax, Robert Künnemann, Sven Tangermann, and Michael Backes. 2019. How to Wrap it up-A Formally Verified Proposal for the use of Authenticated Wrapping in PKCS# 11. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*.

[11] Danny Dolev and Andrew Yao. 1983. On the security of public key protocols. *IEEE Transactions on information theory* 29, 2 (1983), 198–208.

[12] Jason A Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel.. In *NDSS*.

[13] Jason A Donenfeld and Kevin Milner. 2017. *Formal verification of the wireguard protocol*. Technical Report. Technical Report.

[14] P. Eronen and J. Korhonen. 2006. Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol. RFC 4739 (Experimental). https://doi.org/10.17487/RFC4739

[15] P. Eronen, H. Tschofenig, and Y. Sheffer. 2010. An Extension for EAP-Only Authentication in IKEv2. RFC 5998 (Proposed Standard). https://doi.org/10.17487/RFC5998

[16] Santiago Escobar, Catherine Meadows, and José Meseguer. 2009. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*. Springer, 1–50.

[17] Guillaume Girol, Lucca Hirschi, Ralf Sasse, Dennis Jackson, Cas Cremers, and David Basin. 2020. A spectral analysis of noise: a comprehensive, automated, formal analysis of Diffie-Hellman protocols. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1857–1874.

[18] D. Harkins and D. Carrel. 1998. The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard). https://doi.org/10.17487/RFC2409 Obsoleted by RFC 4306, updated by RFC 4109.

[19] Tobias Heider. 2019. Towards a Verifiably Secure Quantum Resistant Key Exchange in IKEv2. http://mnm-team.org/pub/Diplomarbeiten/heid19

[20] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. 2014. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296 (Internet Standard). https://doi.org/10.17487/RFC7296 Updated by RFCs 7427, 7670, 8247.

[21] C. Kaufman (Ed.). 2005. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard). https://doi.org/10.17487/RFC4306 Obsoleted by RFC 5996, updated by RFC 5282.

[22] S. Kent. 2005. IP Authentication Header. RFC 4302 (Proposed Standard). https://doi.org/10.17487/RFC4302

[23] S. Kent. 2005. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard). https://doi.org/10.17487/RFC4303

[24] S. Kent and K. Seo. 2005. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard). https://doi.org/10.17487/RFC4301 Updated by RFCs 6040, 7619.

[25] T. Kivinen. 2016. Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation. RFC 7815 (Informational). https://doi.org/10.17487/RFC7815

[26] Hugo Krawczyk. 2003. SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Annual International Cryptology Conference*. Springer, 400–425.
[27] Gavin Lowe. 1997. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*. IEEE, 31–43.
[28] Robert J. McEliece. 1978. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report* 44 (1978), 114–116.
[29] Catherine Meadows. 1999. Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*. IEEE, 216–231.
[30] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 696–701.
[31] S Moedersheim, PH Drielsma, et al. 1997. AVISPA Project Deliverable D6. 2: Specification of the Problems in the High-Level Specification Language (2003).
[32] E. Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard). https://doi.org/10.17487/RFC8446
[33] Valery Smyslov. 2019. *Intermediate Exchange in the IKEv2 Protocol*. Internet-Draft draft-ietf-ipsecme-ikev2-intermediate-03. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-intermediate-03 Work in Progress.
[34] The Tamarin Team. 2016. *Tamarin-Prover Manual*. https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf
[35] C. Tjhai, M. Tomlinson, grbartle@cisco.com, Scott Fluhrer, Daniel Van Geest, Oscar Garcia-Morchon, and Valery Smyslov. 2020. *Multiple Key Exchanges in IKEv2*. Internet-Draft draft-ietf-ipsecme-ikev2-multiple-ke-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-multiple-ke-00 Work in Progress.
[36] Trevor Perrin. 2018. The Noise protocol framework. https://noiseprotocol.org/noise.html
[37] Tristan Ninet. 26.06.2020. *Formal verification of the Internet Key Exchange (IKEv2) security protocol*. Ph.D. Dissertation. Université Rennes 1. https://tel.archives-ouvertes.fr/tel-02882167/

A TAMARIN-CODE FOR THE VERIFICATION OF IKEV2-SECURITY

A.1 Tamarin Basics

The following Listing states the name of the theory that shall be proven ("IKEv2") and which built-in functions provided by Tamarin shall be used, and restricts the equation function.

```
theory IKEv2
begin

builtins: asymmetric-encryption,
    diffie-hellman, hashing, signing,
    symmetric-encryption
functions: hmac/2

/* Whenever a Eq action occurs, the two
    arguments must be equal */
restriction Eq_check_succeed: "All x y #i.
    Eq(x,y) @ i ==> x = y"

// IKEv2 Protocol

// <INSERT RULES HERE>

// <INSERT LEMMATA HERE>

end
```

A.2 Creating the identity

A static-key-pair for each identity can be created using the following code;
$\sim sk$ denotes the private share and $pk(\sim sk)$ the public share.

```
/* Static Key generator: Keys are bound to
    an ID i, which is not public */

rule generate_static:
[ Fr(~sk) ]                 // < input: fresh
    variable ~sk

--[GenStatic(pk(~sk))]->    // < action fact:
    static key
                            //   generated
                                from ~sk

[ !PrivKey($I, ~sk)         // < $I's priv.
    key is ~sk
, !PubKey($I, pk(~sk))      // < $I's pub. key
     is pk(~sk)
, Out(pk(~sk)) ]            // < pk(~sk) is
    published
```

A.3 The attacker model in Tamarin

In accordance with the attacker model described in Section 3, the following Listing defines the two functions "reveal_static" and "reveal_dh".

```
/* key revelations defined in the adversary
    model */

rule reveal_static:
[ !PrivKey($I, sk) ]    // < If $I's private
     key sk
                        // was generated
                            before
--[RevSk($I)]->         // < then upon use
    of this rule,
                        // the action fact "
                            RevSK($I)" is
                            registered
[ Out(sk) ]             // < and sk is
    published

rule reveal_dh:
[ !DHtoReveal($I, k) ]  // < If $I's
    ephemeral key k
                        // was generated
                            before
--[RevDH($I)]->         // < then upon use
    of this rule,
```

```
                        // the action fact "
                            RevDH($I)" is
                            registered
[ Out(k) ]              // < and k is
    published
```

A.4 The IKEv2 State Machine in Tamarin

In the following, the implementation of the IKEv2 state machine is presented.

A.4.1 IKE_SA_INIT:. The first exchange phase in IKEv2; modeled as the states IKE_SA_INIT_I and IKE_SA_INIT_R.

```
rule IKE_SA_INIT_I: // I initiates
    communication and sends his DH-share
let
epI = 'g'^~eI    // < Initiator's DH-share
msg1 = <~spiI, 'IKE_SA_INIT', '1', 'i', epI,
     ~nI>
         // ^ 'IKE_SA_INIT', '1', 'i' are
             message metadata
in
[ Fr(~nI)        // < Initiator Nonce
, Fr(~spiI)      // < Initiator SPI (session
    id)
, Fr(~eI) ]      // < Initiator ephemeral key
-->
[ Out(msg1)
, StateInitI($I, $R, ~spiI, ~eI, ~nI, msg1)
      // ^ internal state of Initiator is
          registered
, !DHtoReveal($I, ~eI) ]    // < $I's
    ephemeral key is ~eI
```

```
rule IKE_SA_INIT_R: // R agrees and responds
     with his DH-share
let
epR = 'g'^~eR              // < Responder's
    DH-share
k = epI^~eR                // < DH-key,
    calculated with eR
keymat = h(<nI, ~nR>, k)  // < shared key

msg1 = <spiI, 'IKE_SA_INIT', '1', 'i', epI,
    nI>
  // ^ initiator's first message
msg2 = <spiI, ~spiR, 'IKE_SA_INIT', '1', 'r'
    , epR, ~nR>
  // ^ response
in
[ In(msg1)  // < If message 1 was published,
     and
            // given fresh values:
, Fr(~nR)   // < Responder Nonce,
, Fr(~spiR) // < Responder SPI, and
, Fr(~eR) ] // < Responder DH private share

--[ Agreed(<spiI, ~spiR>, $R, 'responder', <
    nI, ~nR>, keymat)]->
  // upon use of this rule, the action fact
      above is registered
  // which states that $R in the role of the
      responder has agreed
  // on the key 'keymat' in the session <
      spiI, spiR>  with nonces <nI, nR>
[ Out(msg2)             // msg2 is published
, StateInitR($I, $R, spiI, ~spiR, nI, ~nR,
    keymat, msg1, msg2, epI, epR)
     // ^ internal state of Responder is
         registered
, !DHtoReveal($R, ~eR) ]   // < $R's
    ephemeral key is ~eR
```

A.4.2 IKE_AUTH exchange: The second exchange phase in IKEv2; modeled as the states IKE_SA_AUTH_I and IKE_SA_AUTH_R. Messages are now encrypted.

```
rule IKE_AUTH_I: // I agrees and
    authenticates himself
let
k = epR^eI              // < DH-key,
    calculated with eI
keymat = h(<nI, nR>, k) // shared key

signed_octets = <msg1, nR, h(~idI, keymat)>
auth_pl = sign(signed_octets, skI)
    // ^ authentication payload
encr_pl = senc{<~idI, auth_pl, pkI>}keymat
   // ^ encrypted payload
integ_I = hmac(<spiI, spiR, 'IKE_AUTH', '2',
     'i', encr_pl>, keymat)
   // ^ hash for integrity protection

msg2 = <spiI, spiR, 'IKE_SA_INIT', '1', 'r',
     epR, nR>
   // ^ R's init message
msg3 = <spiI, spiR, 'IKE_AUTH', '2', 'i',
    encr_pl, integ_I> // < I's auth message

in
[ In(msg2)            // < If msg2 was
    published,
, StateInitI($I, $R, spiI, eI, nI, msg1)
  // ^ and if $I has the this internal state
      ,
, !PrivKey($I, skI)   // < $I's priv. key is
     skI, and
```

```
, !PubKey($I, pkI)     // < $I's pub. key is pkI, and for
, Fr(~idI) ]           // < Initiator identity

--[ Agreed(<spiI, spiR>, $I, 'initiator', <nI, nR>, keymat)]->
  // upon use of this rule, this action fact is registered
  // which states that $I in the role of the initiator
  // has agreed on the key 'keymat' in the session <spiI, spiR>  with nonces <nI, nR>

[ StateAuthI($I, $R, ~idI, spiI, spiR, nI, nR, keymat, msg1, msg2, 'g'^eI, epR)
  // ^ Initiator's new internal state is registered
, Out(msg3) ]     // < and msg3 is published
```

```
rule IKE_AUTH_R: // R authenticates himself and thus completes the mandatory part of the protocol
let
// Initiator's authentication message
signed_octets_I = <msg1, nR, h(idI, keymat)>
encr_pl_I = senc{<idI, auth_pl_I, pkI>}keymat
integ_I = hmac(<spiI, spiR, 'IKE_AUTH', '2', 'i', encr_pl_I>, keymat)
msg3 = <spiI, spiR, 'IKE_AUTH', '2', 'i', encr_pl_I, integ_I>

// Responder's authentication message
signed_octets_R = <msg2, nI, h(~idR, keymat)>
auth_pl_R = sign(signed_octets_R, skR)
encr_pl_R = senc{<~idR, ~spiC, auth_pl_R, pkR>}keymat
integ_R = hmac(<spiI, spiR, 'IKE_AUTH', '2', 'r', encr_pl_R>, keymat)
msg4 = <spiI, spiR, 'IKE_AUTH', '2', 'r', encr_pl_R, integ_R>
in
[ In(msg3)              // < If msg3 was published,
, StateInitR($I, $R, spiI, spiR, nI, nR, keymat, msg1, msg2, epI, epR)
  // ^ and if $R has the this internal state ,
```

```
, !PrivKey($R, skR)    // < $R's priv. key is skR,
, !PubKey($R, pkR)     // < $R's pub. key is pkR, and
, !PubKey($I, pkI)     // < $I's pub. key is pkI, and for
, Fr(~idR)             // < Responder identity
, Fr(~spiC) ]            // < ChildSA session id

--[ Eq(verify(auth_pl_I, signed_octets_I, pkI), true)
  , Completed(<spiI, spiR>, $R, 'responder', $I, keymat)
  , IdentityLearnt(~idR) ]->
  // upon use of this rule, 3 action facts are registered
  // which state that I's signature was verified,
  // that R completed a run of the protocol in the responder role with peer $I
  // in the session <spiI, spiR> with shared key 'keymat'
  // and that a peer learnt R's identity

[ ChildSAR($I, $R, ~spiC, hmac(keymat, <nI, nR>))
  // ^ action fact: Responder derived ChildSA values
, Out(msg4) ] // and msg4 is published
```

```
rule IKE_AUTH_COMPLETE: // I completes mandatory part of the protocol & sends a test-message in the new ChildSA
let
// Responder's authentication message
signed_octets_R = <msg2, nI, h(idR, keymat)>
encr_pl_I = senc{<idR, spiC, auth_pl_R, pkR>}keymat
integ_I = hmac(<spiI, spiR, 'IKE_AUTH', '2', 'r', encr_pl_I>, keymat)
msg4 = <spiI, spiR, 'IKE_AUTH', '2', 'r', encr_pl_I, integ_I>

ck = hmac(keymat, <nI, nR>)           // < ChildSA key
mTest = <'0', senc('test', ck),
   hmac(<'0', senc('test',ck)>, ck)> // < test message, encrypted and integrity protected with ck
in
```

```
[ In(msg4)   // If this message was published,
, StateAuthI($I, $R, idI, spiI, spiR, nI, nR, keymat, msg1, msg2, epI, epR)
                // ^ if this is I's internal state, and
, !PubKey($R, pkR) // if this is $R's public static key ]
--[ Eq(verify(auth_pl_R, signed_octets_R, pkR), true)
,   Completed(<spiI, spiR>, $I, 'initiator', $R, keymat)
,   IdentityLearnt(idI)
,   IKeys($I, $R, spiC, ck) ]->
  // upon use of this rule, 4 action facts are registered
  // which state that R's signature was verified,
  // that I completed a run of the protocol in the initiator role with peer $R
  // in the session <spiI, spiR> with shared key 'keymat'
  // and that a peer learnt R's identity and I registered ck as ChildSA-key
[ Out(mTest) ]    // < Test-message is published

rule ChildSA_Confirm_R:
[ ChildSAR($I, $R, spiC, ck) ] // < If there is a ChildSA
, In(<'0', senc('test', ck),
     hmac(<'0', senc('test',ck)>, ck)>)
      // ^ and a valid test message was published for it
--[ RConfirm($I, $R, spiC, ck)]-> // < then upon use of this rule, the session registers that R has received a message valid under the established ChildSA key.
[ ]
```

A.5 IKE_INTERMEDIATE

```
rule IKE_INTERMEDIATE_I: // I's second key exchange message
let
msg2 = <spiI, spiR, 'IKE_SA_INIT', '1', 'r', epR, nR>

k = epR^eI
keymat = h(<nI, nR>, k)

// I encrypts and integrity-protects the message with the previously exchanged key
encr_pl_I = senc{<'g'^~peI, ~nI2>}keymat
integ_I = hmac(<spiI, spiR, 'IKE_INTERMEDIATE', '2', 'i', encr_pl_I>, keymat)
msgINT = <spiI, spiR, 'IKE_INTERMEDIATE', '2', 'i', encr_pl_I, integ_I>

in
[ In(msg2)
, Fr(~peI)
, Fr(~nI2)
, StateInitI($I, spiI, eI, nI, msg1)
]
--[ Agreed(<spiI, spiR>, $I, 'initiator', <nI, nR>, keymat)
, INTERMEDIATE_I($I, $R) ]-> //< action fact : INTERMEDIATE exchange happened between $I and $R

[ Out(msgINT)
, StateIntermI($I, $R, spiI, spiR, nI, nR, ~nI2, keymat, msg1, msg2, msgINT, 'g'^eI, epR, ~peI)
, !DHQtoReveal($I, ~peI) ] // < $I's quantum-safe key is peI
```

```
rule IKE_INTERMEDIATE_R:  // R's second key exchange message
let
// I's second key exchange message, encrypted and integrity-protected with the previously exchanged key
encr_pl_I = senc{<pepI, nI2>}keymat_old
integ_I = hmac(<spiI, spiR, 'IKE_INTERMEDIATE', '2', 'i', encr_pl_I>, keymat_old)
msgINT = <spiI, spiR, 'IKE_INTERMEDIATE', '2', 'i', encr_pl_I, integ_I>

pq = pepI^~peR
keymat = h(keymat_old, <pq, nI2, ~nR2>)

// R encrypts and integrity-protects the message with the previously exchanged key
encr_pl_R = senc{<'g'^~peR, ~nR2>}keymat_old
integ_R = hmac(<spiI, spiR, 'IKE_INTERMEDIATE', '2', 'r', encr_pl_R>, keymat_old)
```

```
msgINT2 = <spiI, spiR, 'IKE_INTERMEDIATE', '2', 'r', encr_pl_R, integ_R>
in
[ In(msgINT)
, Fr(~peR)
, Fr(~nR2)
, StateInitR($I, $R, spiI, spiR, nI, nR, keymat_old, msg1, msg2, epI, epR)
]
--[ Agreed(<spiI, spiR>, $R, 'responder', <nI2, ~nR2>, keymat)]->

[ Out(msgINT2)
, StateIntermR($I, $R, spiI, spiR, nI, nR, nI2, ~nR2, keymat_old, keymat,
msg1, msg2, msgINT, msgINT2, epI, epR, pepI, 'g'^~peR)
, !DHQtoReveal($R, ~peR) ] < $R's quantum-safe key is peR
```

A.6 Lemmata

In the following, the implementation of proven Lemmata according to Section 3 is presented.

A.6.1 Correctness. To prove the correctness of the model, there are two lemmata: the first states that it is possible for an Initiator and a Responder to successfully establish a shared key with each other, the second states that it is possible for them to establish two different shared keys with each other.

```
/* There is a set of parameters such that I and R complete a run of the protocol with each other and agree on a shared key. */
lemma exists_session: exists-trace
"Ex I R spi #i #j keymat.
Completed(spi, I, 'initiator', R, keymat) @ #j
& Completed(spi, R, 'responder', I, keymat) @ #i
& #i < #j"  // no loss of generality*

/* There are two sets of parameters such that I and R complete corresponding runs of the protocol with each other and agree on two different shared keys. */
lemma exists_two_sessions: exists-trace
"Ex I R spi spi2 ck ck2 #i #j #i2 #j2 .
IKeys(I, R, spi, ck) @ #i
& RConfirm(I, R, spi, ck) @ #j
& #i < #j   // no loss of generality*
& IKeys(I, R, spi2, ck2) @ #i2
& RConfirm(I, R, spi2, ck2) @ #j2
& #i2 < #j2   // no loss of generality*
& not (ck=ck2)"
```

* This restriction helps the automatic prover find such a trace.

A.6.2 Authentication. Authentication is split in the Lemmata Aliveness, Weak Agreement_I, Weak Agreement_R, Agreement_I and Agreement_R, as explained in Section 3:

```
lemma aliveness[use_induction]:
"All spi A B keymat role #i .
Completed(spi, A, role, B, keymat) @ #i
& not (Ex #k . RevSk(B) @ k)
==> (Ex spi2 role2 peer nonce keymat2 #j .
    Agreed(spi2, B, role2, peer, nonce, keymat2) @ #j
& #j < #i)"
```

```
lemma weak_agreement_i[use_induction]:
"All spi I R keymat #i .
Completed(spi, I, 'initiator', R, keymat) @ #i
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDH(I) @ k)
& not (Ex #k . RevDH(R) @ k)
==> (Ex spi2 role nonce keymat2 #j . Agreed(spi2, R, role, I, nonce, keymat2) @ #j
& #j < #i)"
```

```
lemma weak_agreement_r[use_induction]:
"All spi I R keymat #i .
Completed(spi, R, 'responder', I, keymat) @ #i
& not (Ex #k . RevSk(I) @ k)
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDH(I) @ k)
& not (Ex #k . RevDH(R) @ k)
==> (Ex spi2 role nonce keymat2 #j . Agreed(spi2, I, role, R, nonce, keymat2) @ #j
& #j < #i)"
```

```
lemma agreement_i[use_induction]:
"All spi I R keymat #j .
Completed(spi, I, 'initiator', R, keymat) @ #j
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDH(I) @ k)
& not (Ex #k . RevDH(R) @ k)
==> (Ex spi2 #k .
Completed(spi2, R, 'responder', I, keymat) @ #k)"
```

```
lemma agreement_r[use_induction]:
"All spi I R keymat #j .
```

```
Completed(spi, R, 'responder', I, keymat) @ #j
& not (Ex #k . RevSk(I) @ k)
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDH(I) @ k)
& not (Ex #k . RevDH(R) @ k)
==> (Ex spi2 #k .
Completed(spi2, I, 'responder', R, keymat) @ #k)"
```

A.6.3 Session Uniqueness. The following lemma proves Session uniqueness:

```
lemma session_uniqueness:
"All I R spi spi2 keymat role #j #l.
Completed(spi, I, role, R, keymat) @ #j
& Completed(spi2, I, role, R, keymat) @ #l
==> (#j = #l)"
```

A.6.4 Consistency. The following lemma proves Consistency:

```
lemma consistency:
"All spi I R keymat keymat2 #i #j .
Completed(spi, I, 'initiator', R, keymat) @ #i
& Completed(spi, R, 'responder', I, keymat2) @ #j
& not (Ex #k . RevSk(R) @ #k)
==> (keymat=keymat2)"
```

A.6.5 Key Secrecy. The following lemma proves Key secrecy:

```
lemma key_secrecy[reuse]:
"All spi I R role keymat #j .
Completed(spi, I, role, R, keymat) @ #j
& not (Ex #m . RevSk(R) @ #m)
& not (Ex #m . RevDH(I) @ #m)
& not (Ex #m . RevDH(R) @ #m)
==> not (Ex #m . K(keymat) @ #m)"
```

A.6.6 Identity Hiding. The following lemma proves Identity Protection:

```
lemma identity_hiding_R:
"All spi I R keymat id #i .
Completed(spi, R, 'responder', I, keymat) @ #i
& IdentityLearnt(id) @ #i
& not (Ex #k . RevSk(I) @ #k)
& not (Ex #k . RevDH(I) @ #k)
& not (Ex #k . RevDH(R) @ #k)
==> not (Ex #j .  K(id) @ #j)"
```

A.7 Additional code for verifying quantum-safe IKEv2

A.7.1 Revelation of quantum-resistant key. The following lemma allows the revelation of quantum-resistant DH-keys.

```
rule reveal_dhq:
[ !DHQtoReveal($I, k) ]
--[RevDHQ($I)]->
[ Out(k) ]
```

A.7.2 Affected lemmata. As stated in Section 5.2, the lemmata for Weak Agreement, Agreement, Key Secrecy, and Identity Protection are only achieved if the DH-values remain secret, and therefore need to be updated for modeling a quantum-based attacker.

```
lemma weak_agreement_i[use_induction]:
"All spi I R keymat #i .
Completed(spi, I, 'initiator', R, keymat) @ #i
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDHQ(I) @ k)
& not (Ex #k . RevDHQ(R) @ k)
==> (Ex spi2 role nonce keymat2 #j . Agreed(spi2, R, role, I, nonce, keymat2) @ #j
& #j < #i)"
```

```
lemma weak_agreement_r[use_induction]:
"All spi I R keymat #i .
Completed(spi, R, 'responder', I, keymat) @ #i
& not (Ex #k . RevSk(I) @ k)
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDHQ(I) @ k)
& not (Ex #k . RevDHQ(R) @ k)
==> (Ex spi2 role nonce keymat2 #j . Agreed(spi2, I, role, R, nonce, keymat2) @ #j
& #j < #i)"
```

```
lemma pq_agreement_i[use_induction]:
"All spi I R keymat #j .
Completed(spi, I, 'initiator', R, keymat) @ #j
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDHQ(I) @ k)
& not (Ex #k . RevDHQ(R) @ k)
==> (Ex spi2 #k .
Completed(spi2, R, 'responder', I, keymat) @ #k)"
```

```
lemma pq_agreement_r[use_induction]:
"All spi I R keymat #j .
Completed(spi, R, 'responder', I, keymat) @ #j
& not (Ex #k . RevSk(I) @ k)
& not (Ex #k . RevSk(R) @ k)
```

```
& not (Ex #k . RevDHQ(I) @ k)
& not (Ex #k . RevDHQ(R) @ k)
==> (Ex spi2 #k .
Completed(spi2, I, 'responder', R, keymat) @
    #k)"

lemma session_uniqueness:
"All I R spi spi2 keymat role #i #j .
Completed(spi, I, role, R, keymat) @ #i
& Completed(spi2, I, role, R, keymat) @ #j
==> (#i = #j)"

lemma pq_key_secrecy[reuse]:
"All spi I R role keymat #i .
Completed(spi, I, role, R, keymat) @ #i
& not (Ex #j . RevSk(R) @ #j)
& not (Ex #j . RevDHQ(I) @ #j)
& not (Ex #j . RevDHQ(R) @ #j)
==> not (Ex #j . K(keymat) @ #j)"

lemma pq_identity_hiding_R:
"All spi R I keymat id #i .
Completed(spi, R, 'responder', I, keymat) @
    #i
& IdentityLearnt(id) @ #i
& not (Ex #k . RevSk(I) @ #k)
& not (Ex #k . RevDHQ(I) @ #k)
& not (Ex #k . RevDHQ(R) @ #k)
==> not (Ex #j .  K(id) @ #j)"
```

A.8 Alternative code for authentication lemmata where "has been running the protocol" is interpreted as "has completed the protocol"

A.8.1 Variants for standard IKEv2.

```
lemma aliveness_i[use_induction]:
"All spi I R keymat #i .
Completed(spi, I, 'initiator', R, keymat) @
    #i
& not (Ex #k . RevSk(R) @ k)
==> (Ex spi2 peer role keymat2 #j .
    Completed(spi2, R, role, peer, keymat2)
    @ #j
& #j < #i)"

lemma aliveness_r[use_induction]:
"All spi I R keymat #i .
Completed(spi, R, 'responder', I, keymat) @
    #i
& not (Ex #k . RevSk(I) @ k)
==> (Ex spi2 peer role keymat2 #j .
    Completed(spi2, I, role, peer, keymat2)
    @ #j
& #j < #i)"

lemma weak_agreement_i[use_induction]:
"All spi I R keymat #i .
Completed(spi, I, 'initiator', R, keymat) @
    #i
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDH(I) @ k)
& not (Ex #k . RevDH(R) @ k)
==> (Ex spi2 role keymat2 #j . Completed(
    spi2, R, role, I, keymat2) @ #j
& #j < #i)"

lemma weak_agreement_r[use_induction]:
"All spi I R keymat #i .
Completed(spi, R, 'responder', I, keymat) @
    #i
& not (Ex #k . RevSk(I) @ k)
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDH(I) @ k)
& not (Ex #k . RevDH(R) @ k)
==> (Ex spi2 role keymat2 #j . Completed(
    spi2, I, role, R, keymat2) @ #j
& #j < #i)"
```

A.8.2 Variants for quantum-safe IKEv2.

```
lemma pq_weak_agreement_i[use_induction]:
"All spi I R keymat #i .
Completed(spi, I, 'initiator', R, keymat) @
    #i
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDHQ(I) @ k)
& not (Ex #k . RevDHQ(R) @ k)
==> (Ex spi2 role keymat2 #j . Completed(
    spi2, R, role, I, keymat2) @ #j
& #j < #i)"

lemma pq_weak_agreement_r[use_induction]:
"All spi I R keymat #i .
Completed(spi, R, 'responder', I, keymat) @
    #i
& not (Ex #k . RevSk(I) @ k)
& not (Ex #k . RevSk(R) @ k)
& not (Ex #k . RevDHQ(I) @ k)
& not (Ex #k . RevDHQ(R) @ k)
==> (Ex spi2 role keymat2 #j . Completed(
    spi2, I, role, R, keymat2) @ #j
& #j < #i)"
```