FORMAL VERIFICATION OF THE POST-QUANTUM SECURITY PROPERTIES OF IKEV2 PPK (RFC 8784) USING THE TAMARIN PROVER

SOPHIE STEVENS

Arqit Limited, London

Abstract. The Internet Key Exchange version 2 (IKEv2) (RFC 7296) is a component of IPsec used to authenticate two parties (the initiator and responder) to each other and to establish a set of security parameters for the communications. The security parameters include secret keys to encrypt and authenticate data as well as the negotiation of a set of cryptographic algorithms. The core documentation uses exclusively Diffie-Hellman exchanges to agree the security information. However, this is not a quantum-secure option due to the ability of Shor's algorithm [14] to break the security assumption underlying the Diffie-Hellman. A post-quantum solution is to include a preshared key in the exchange, as proposed by the extension RFC 8784 [6]; assuming that this preshared key has sufficient entropy, the keys created in the IKEv2 exchange will be resistant to a quantum computer. In this paper, we investigate the security claims of RFC 8784 using formal verification methods. We find that keys created using the preshared key are secret from an adversary. However, certain authentication properties of the protocol that are weakened under the assumption that Diffie-Hellman is insecure are not recovered using the preshared key.

1. Introduction

The Internet Key Exchange version 2 (IKEv2) [8] is a component of IPsec used to authenticate two parties (the initiator and responder) to each other and to establish a set of security parameters for the communications. The security parameters include secret keys to encrypt and authenticate data as well as the negotiation of a set of cryptographic algorithms. The core documentation uses exclusively Diffie-Hellman exchanges to agree the security information. However, this is not a quantum-secure option due to the ability of Shor's algorithm [14] to break the security assumption underlying the Diffie-Hellmann (specifically, the ability of Shor's algorithm to detect hidden subgroups in abelian groups renders the Discrete Logarithm Problem solvable in polynomial time). One solution is to include a preshared key in the exchange, as proposed by the extension RFC 8784 [6]; assuming that this preshared key (known also as a pre-positioned key and its acronym PPK) has sufficient entropy (e.g. 256 bits), the keys created in the IKEv2 exchange will be resistant to a quantum computer.

E-mail address: sophie.stevens@arqit.uk.

Date: November 13, 2023.

In this paper, we investigate the security claims of RFC 8784 using formal verification methods, namely the Tamarin tool [19]. Our code [18] is publicly accessible on Github [1]. The Tamarin prover models a Dolev-Yao adversary [5]; we assume additionally that this adversary is able to recover x from the Diffie-Hellmann modular exponential g^x. We find that keys created using the preshared key are secret from an adversary. This includes keys that are created in the Create Child SA and Rekey SA sections of the protocol. In the initial exchange, a set of authentication and encryption keys are created without use of the PPK (using only Diffie-Hellmann methods) and these keys are known to our adversary. We study also the authentication properties of the protocol, since authentication of the peers is one of the desired security properties of the IKEv2 protocol. We study authentication following the formal treatment of Lowe [9] – see Section 3.2 for a more thorough description of this. We find that, in the default implementation of the protocol, authentication properties of certain parts of the protocol are fully compromised under the assumption that Diffie-Hellman is insecure and are not recovered using the preshared key. We investigate a solution that is briefly described in the documentation (namely that 'IKE SAs **MUST** rekey the IKE SA before the sensitive information is sent'); we find that this remediation reintroduces authentication properties to the protocol. We note that this solution is actually introduced as a defence-in-depth mitigation to provide secrecy in the case of an attacker having the ability to both decrypt the initial IKE SA and access to a quantum computer; authentication does not appear to be addressed.

2. IKEv2

2.1. **Previous analyses of IKEv2.** The AVISPA[2] platform [1] provided the first formal analysis of IKEv2 in 2003. It aimed to check the authentication and secrecy goals of IKEv2. Further analysis was carried out by Cremers using the Scyther tool [3]. Both studies confirmed weaknesses in authentication for IKEv2 when digital signatures are used for authentication purposes. In this work, we assume that authentication is achieved via a preshared key, where this discovered weakness is irrelevant. Cremers found also a reflection attack; this attack is later disputed by Ninet et al. [10], who point out that Cremers' header does not include the specified *Initiator* and *Responder* flags. These flags are integrity-protected during the authentication payload. Ninet et al. use the tool SPIN [10] to check the secrecy, aliveness and weak agreement of three aspects of IKEv2 protocol. They found that a key exchange established with a preshared key, and the creation of a Child SA satisfied the desired secrecy property as well as the authentication properties of aliveness and weak agreement. (The third sub-protocol that the team analysed was IKEv2 using a digital signature, whereby they found that the authentication property of weak agreement is not satisfied.)

Ninet et al. [11] introduce a *deviation attack*. This requires initiator parties to authenticate themselves using digital signatures. The deviation attack works is by intercepting traffic between an initiator and responder and sending these exchanges to a victim (tricking the victim into believing that it is the intended responder). This results in a memory exhaustion as the victim is forced to create many unfinished connections with the initiator. This attack is viable when preshared keys

[1]All code is available here: https://github.com/sophie-arqit/ikev2-rfc8784-tamarin-analysis

[2]AVISPA is an acronym for Automated Validation of Internet Security-sensitive Protocols and Applications. This tool is now deprecated.

are used for authentication under the violation of the assumption that a preshared key is shared only between two parties.

With a complementary goal to this note, Gazdag et al. [7] provide a formal analysis of IKEv2's post-quantum extension. This is distinct to the PPK extension that we study here. Explicitly, Gazdag et al. study a minimal version of IKEv2 [8] and consider the case when an adversary has access to a quantum computer. In this situation, Diffie-Hellman is no longer secure and thus they formally prove that neither is IKEv2. They analyse a proposed extension in which Diffie-Hellman key-exchange is enhanced by the use of multiple quantum-resistant key exchanges [16, 20]. Under the assumption that the quantum-resistant key exchange algorithm used is unbreakable, they show that this extension recovers the security properties of IKEv2.

An excellent resource on the formal verification of IKEv2 is the doctoral thesis of Ninet [12]. In this thesis, Ninet extensively analyses IKEv2 using three formal verification tools: SPIN [15], ProVerif [2] and Tamarin [19]. He consistently finds across the three platforms that IKEv2 achieves the full desired secrecy and authentication properties for the initial authenticated exchange when a preshared key is used for authentication – if instead a digital signature is used for authentication, then certain authentication properties no longer hold. Furthermore, he analyses the Create Child SA exchange and finds that although secrecy is preserved, that certain authentication properties are not. Specifically, non-injective agreement (and thus the stronger injective agreement) property of authentication is compromised: the initiator completes the exchange (creating a Child SA), and, although it interacted with the responder, the two agents do not agree on all intermediate data values.

2.2. **Overview of IKEv2.** The goal of IKEv2 is for an initiator and responder to agree on a cryptographic algorithm and key to exchange encrypted communication. The data describing the algorithms that they may use and the associated keys is called the Security Association (SA). Having established an SA, IKEv2 enables the parties to create further SAs (Child SAs) and to rekey both the SA and the Child SA. We think of this protocol as three parts: (i) the initial communication stage (SA Init) where the shared key is agreed, (ii) the authentication stage (SA AUTH), where the messages and actors in previous SA Init stage are authenticated, (iii) the CreateChild stage that includes the ability to rekey the current SA, to create a Child SA and to rekey a Child SA. We now explain the three stages in more detail.

2.2.1. *SA Init.* The initiator sends a header containing: the Security Parameter Index (SPI), the descriptor of the phase of the protocol and flags containing the information that this message comes from an initiator and the Message ID. The initiator then also sends a Security Association. This is a list of cryptographic functions supported by the initiator. The initiator picks a private Diffie-Hellman value and sends the key exchange information of the corresponding public value. Finally the initiator picks a random nonce.

The responder replies with a similar header; the responder's header must include the Security Parameter Index of both itself and the initiator. The responder sends a Security Association in the form of a cryptographic algorithm (or a set of algorithms used for different purposes) chosen from

the initiator's supported set. The responder chooses a private Diffie-Hellman value and sends the public value and finally also chooses a nonce. If the responder wishes

Additionally, in the PPK extension, both the initiator and receiver send a notification `N(USE_PPK)` to indicate that they both wish to use a preshared key. We consider only the instance where both parties want to use a PPK.

```
HDR(SPIi, IKE_SA_INIT, Flags: Initiator, Message ID=0),
SAi1, KEi, Ni, N(USE_PPK) -->
                    <--  HDR(SPIi, SPIir, IKE_SA_INIT, Flags: Response, Message ID=0),
                           SAr1, KEr, Nr, N(USE_PPK)
```

At the end of this exchange, both parties have agreed on which algorithms to use, a key and they have agreed to use a PPK.

In IKEv2 there are seven keys that are used: `SK_d` is used to rekey; `SK_ai` and `SK_ar` provide integrity for messages from the initiator and responder respectively; `SK_ei` and `SK_er` are used to encrypt messages sent from the initiator and responder respectively; `SK_pi` and `SK_pr` are used for the AUTH payload from the initiator and responder respectively. These keys are generated as the output of a seeded pseudorandom function (PRF) `prf+` that is constructed from a PRF established in the Security Association. We do not describe `prf+` here. The seed is

```
SKEYSEED = prf(Ni | Nr, g^ir)
```

where g is the universal Diffie-Hellman generator (established during the SA) and g^{ir} is calculated from the private and public key exchange information.

From SKEYSEED, the keys are generated as the output of `prf+`:

```
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr'}
                     = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr)
```

Once it has been agreed which PPK is to be used, the `SK_d`, `SK_pi` and `SK_pr` values are updated as follows:

```
SK_d = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr').
```

2.2.2. *SA AUTH.* In the AUTH exchange, both parties provide mutual information as well as information about Traffic Selectors (TS). A TS contains the protocol to be used (e.g. AES), acceptable (source and/or destination) address ranges for traffic and the port range allowed. Communication of a TS is always encrypted, by either

`SK_ei`

To simplify the model, we do not consider Traffic Selectors in the formal verification of the process. It is within the AUTH exchange that the initiator chooses the PPK to be used to enhance the security of the key. Each PPK is given a unique identity, and it is this identity that is shared.

We describe the AUTH payload for the initiator only, simplified somewhat for descriptive purposes; the responder sends a similar message using values relevant to itself instead. In the below, RealMessage1 corresponds to the *entire* initial message sent by the initiator, inclusive of the header; InitID corresponds to a descriptor of the initiator's private identity and the identity itself. The identification payloads allow peers to assert an identity to each other - a peer may have multiple identities (although an identity may only be used by a single peer).

```
InitiatorSignedOctets = RealMessage1 | Nr | MACedIDForI
MACedIDForI = prf(SK_pi, InitID)
AUTH = prf( prf(Shared Secret, "Key Pad for IKEv2"), <InitiatorSignedOctets>)
```

The Shared Secret is a preshared key from the IKEv2 specification, pre-dating the PPK extension. The identities used in InitID must match the identity associated to this shared secret. It is not specified how this shared secret is distributed, nor whether this shared secret may be the same as the PPK in the later extension. For the purposes of this model, we assume that they are different.

Finally, the SA__AUTH exchange proceeds as follows:

```
HDR, SK{IDi, AUTH, SAi2, TSi, TSr, N(PPK_IDENTITY, PPK_ID)} --->
                      <-- HDR, SK {IDr,  AUTH, SAr2, TSi, TSr, N(PPK_IDENTITY)}
```

The notification `N(PPK_IDENTITY, PPK_ID)` denotes a notification containing the descriptor that it contains a PPK identity, and then the `PPK_ID` itself. The notation `SK{...}` denotes that the payload is both encrypted and authenticated, using the keys `SK_ei` and `SK_ai` for messages from the initiator, and `SK_er` and `SK_ar` for messages from the responder. The inclusion of new SAs – `SAi2` and `SAr2` – permits the peers to propose changing the algorithms used in future exchanges. Each time a new SA is created, it is associated with new keys as described in the next section.

2.2.3. *CreateChild: Rekey, Create Child, Rekey Child.* A Child SA is a set of algorithms and associated set of keys that will be used for the Encapsulating Security Payload of Authentication Header protocols within IPSec. A Child SA can also be used for an IKE exchange; in this situation, the Child SA has the ability to create further Child SAs. The CreateChild exchange is actually three exchanges. Firstly, a Child SA may be created. The initiator sends a Child SA offer, as well as a new nonce and proposed Traffic Selectors. Optionally a new Diffie-Hellman value is also given. We consider the only the case when a new Diffie-Hellman value is given so that the Child SA is endowed with a new key. The responder replies with a chosen SA, a nonce, Traffic Selector agreement and optionally a new Diffie-Hellman value. The exchange is:

```
HDR, SK {SA, Ni, [KEi,]  TSi, TSr} -->
                    <-- HDR, SK {SA, Nr, [KEr,] TSi, TSr}
```

To rekey a (non-child) SA, a similar exchange occurs. New SPI values are exchanged within the new SA payload.

```
HDR, SK {SA, Ni, KEi} -->
                    <-- HDR, SK {SA, Nr, KEr}
```

Finally, to rekey a Child SA, the exchange is again almost identical:

```
HDR, SK {N(REKEY_SA), SA, Ni, [KEi,] TSi, TSr} -->
 <-- HDR, SK {SA, Nr, [KEr,] TSi, TSr}
```

Note that a new Diffie-Hellman value need not be provided to create a new key. The distinction between the exchanges is recognised within the header.

There is additionally an Informational Exchange in IKEv2. This allows a party to report error conditions or to revoke an SA. These exchanges are typically cryptographically protected with the negotiated keys; messages pertaining to an IKE SA in most circumstances must be sent under that IKE SA. Thus, a compromise of the IKE SA is equivalent to a compromise of the Informational Exchange. Some Informational Messages (e.g. notification about protocol incompatibility) may occur outside of an IKE SA if there is no suitable SA; in this case, there is no cryptographic protection. These messages do not directly lead to the creation of new IKE SAs and so we do not further consider the security of Informational Exchange messages.

To generate keying material for Child SAs, including the Child SA created in the SA_AUTH exchange we use:

```
KEYMAT = prf+(SK_d, Ni | Nr).
```

For CREATE_CHILD_SA exchanges including an optional Diffie-Hellman exchange, the keying material is defined as

```
KEYMAT = prf+(SK_d, g^ir (new) | Ni | Nr).
```

The new key to rekey an existing IKE SA is created as follows

```
SKEYSEED = prf(SK_d, g^ir (new) | Ni | Nr)
KEYMAT = prf+(SKEYSEED, Ni|Nr|SPIi|SPIr).
```

KEYMAT should be understood as the output of the prf construction, with the output iteratively generating as many of the key values `{SK_d | SK_ai | SK_ar | SK_ei | SK_er | SK_pi | SK_pr}` as required. In each case, it is the new nonces (and, if relevant, new shared Diffie-Hellman values) in the CreateChild exchange that are used in the key material and the old value of `SK_d`. Unlike in the initial creation of keys, a PPK is not used to strengthen these new keys. This is because the key `SK_d` already includes the PPK and so, as we will demonstrate, the secrecy of future keys is preserved.

2.3. **Post-quantum Preshared Keys extension.** In June 2020 the IETF published the Request for Comments number 8784 [6] (PPK extension) that described how to mix preshared keys in IKEv2

for the purpose of achieving post-quantum security. Its goal was to describe 'an extension of IKEv2 to allow it to be resistant to a quantum computer by using preshared keys'. Without this extension, IKEv2 is vulnerable to the threat of quantum computers; since the security of Diffie-Hellman relies on the hardness of the discrete log problem a quantum computer can solve Diffie-Hellman key exchange problems in polynomial time [14]. Since the keys in IKEv2 are generated by a Diffie-Hellman exchange, it follows that a quantum computer will render the protocol entirely compromised. The extension assumes that the initiator and responder share a list of Post-quantum Preshared Keys (PPKs) with matching identifiers; the protocol is agnostic on how these keys are initiated and these keys are assumed to be configured in an out-of-band mechanism.

An aim of this extension is to minimise the disruption within the IKEv2 protocol. We recall that in IKEv2 that a seed (comprised of a Diffie-Hellman secret value, nonces and other communicated data) generates keys that are used for four purposes: (i) to encrypt exchanges; (ii) to provide integrity-protection on exchanges; (iii) to provide authentication in the AUTH payload; (iv) to derive new keys. These keys are generated in an identical manner in the PPK extension. However, the keys that are used to provide authentication in the AUTH payload and to derive new keys are then updated by hashing the key in question together with the PPK that the initiator and responder have agreed upon. As demonstrated in this note, this provides post-quantum security for new keys and for the authentication process. However, keys that are used to encrypt and provide integrity-protection on exchanges are *not* updated using the PPK, and so this part of the protocol is vulnerable to the effects of a quantum computer.

3. Security model

3.1. **Dolev-Yao adversary.** We consider the Dolev-Yao model for the adversary [5]: an adversary has the ability to block, create, reorder, replay and initiate any communication sent between honest actors. Certain assumptions are made on the security of abstract cryptographic primitives, namely that encryption (both symmetric and asymmetric) is secure, hash functions are one-way, there is a public directory that is secure and cannot be tampered with. As with any model, the Dolev-Yao model has limitations that it is important to understand: we cannot quantify 'how' secure a communication is, or the quantitative effect of a partial leak of secret material.

The only asymmetric cryptography that is relevant to IKEv2 PPK is the Diffie-Hellman key exchange. Since we are concerned with post-quantum security, we additionally endow our adversary with the ability to recover the exponent e of a public Diffie-Hellman value g^e (where g, the generator of the group is a public variable).

3.2. **Security properties.** The security goals of the PPK extension are described within the IKEv2 PPK documentation. We extract the key points directly:

- *The primary goal of this document is to augment the IKEv2 protocol to provide protection against quantum computers without requiring novel cryptographic algorithms.*
- *This document makes the SK_d (and thus also the IPsec KEYMAT and any subsequent IKE SA's SKEYSEED) depend on both the symmetric PPK and the Diffie-Hellman exchange. So, even if the (EC)DH can be trivially solved, the attacker still can't recover any key material*

(except for the SK_{ei}, SK_{er}, SK_{ai}, and SK_{ar} [keys for encryption and integrity-protection for the initiator and responder respectively] *values for the initial IKE exchange)*

- *Note that we do allow an attacker with a quantum computer to rederive the keying material for the initial IKE SA; this was a compromise to allow the responder to select the correct PPK quickly.*
- *A fourth goal is to avoid violating any of the security properties provided by IKEv2.*

The main security goals of IKEv2 are to provide the secrecy of the keying material and mutual authentication between the initiator and responder.

We investigate authentication as defined by the *hierarchy of authentication* introduced by Lowe [9]. He considers successively stronger definitions of authentication, from the weakest authentication goal of aliveness, to the strongest authentication goal of injective agreement. We also investigate the secrecy of the key material. Another desirable feature is that generated key material is unique.

We now define these security properties explicitly, considering a protocol exchange between two actors Agent A and Agent B.

- **Aliveness**: Whenever Agent A has completed the protocol, apparently with Agent B, then Agent B has previously run the protocol (although not necessarily with Agent A).
- **Weak Agreement**: Whenever Agent A has completed the protocol, apparently with Agent B, then Agent B has previously run the protocol, apparently with Agent A. Note that Agent A and Agent B need not agree on the role played by Agent B - for example, they could both believe that they initiated the protocol.
- **Non-injective Agreement**: Whenever Agent A has completed the protocol, apparently with Agent B, then Agent B has previously run the protocol, apparently with Agent A. Additionally, both actors agree on their respective roles and also agree on all intermediate data values. Note that this does not exclude the possibility of Agent B thinking it completed two runs with Agent A. i.e. There is not necessarily an injective relationship between the sessions of Agent A and the sessions of Agent B.
- **Injective Agreement**: Whenever Agent A has completed the protocol, apparently with Agent B, then Agent B has previously run the protocol, apparently with Agent A. Additionally, both actors agree on their respective roles and also agree on all intermediate data values. Finally, each run of Agent A corresponds to a unique run of Agent B.
- **Secrecy**: The adversary does not have access to the secret keys.
- **Uniqueness of Keys**: If Agent A and Agent B create two keys from two different runs of the protocol, then the two keys are different. This assumes that nonces are randomly generated from honest users.

Note that both the initiator or responder can take the role of Agent A. Hence the four authentication properties entail eight checks, with each of Initiator and Responder in the role of Agent A.

4. Methods

4.1. **Tamarin.** The Tamarin prover [19] is a formal verification tool suited to the symbolic modelling of security protocols. An abstraction of the protocol is implemented, with cryptographic primitives (such as symmetric encryption and one-way functions) assumed perfect. In addition, Tamarin provides support for the algebraic properties of Diffie-Hellman key exchange. One is then able to reason about the protocol, asking whether certain properties (expressed in first-order logic) do or do not hold. The Tamarin tool turns the requests ("lemmas") into a constraint solver and, if the program terminates, then the property is shown to hold or not to hold. If the property holds, a proof is provided; if it does not, then a counter-example is found. Tamarin was initially developed by Simon Meier and Benedikt Schmidt with support from David Basin and Cas Cremers.

Tamarin has had notable success with the development of TLS 1.3 [4] – formal verification using Tamarin occurred alongside the development process and so enable a secure-by-design approach.

The built-in adversary within Tamarin is the Dolev-Yao adversary, described in Section 3.1. It is also possible to provide the adversary with additional abilities by introducing new rules. In particular, we endow our adversary with the ability to break Diffie-Hellman by publishing the secret values when they are created. When we consider the security of IKEv2 PPK under the assumption that Diffie-Hellman is *not* broken, then we simply do not give our adversary this ability.

4.2. **Protocol model.** To implement IKEv2 and its PPK extension we make several simplifications in abstracting the model:

- The SA payload specifies the list of algorithms that the initiator would like to support and from which the responder chooses which algorithms to use. Since Tamarin is agnostic as to the choice of algorithms, we model the SA payload as a random variable that is included in payloads that are encrypted and authenticated. This means that the peers must have agreed on the SAs.
- We do not include the Traffic Selector payload.
- We assume that there is a one-to-one relationship between PPKs and PPK_IDs.
- We assume that the initiator and responder have a unique Shared Secret that they use for the identification step.

5. Results

5.1. **Secrecy.** In the initial IKE exchange, seven keys are created: `SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, SK_pr`. The keys `SK_ai, SK_ar, SK_ei, SK_er` are known to the adversary. This is because these keys to not incorporate the PPK and are generated only by Diffie-Hellman values that are insecure in the presence of a quantum computer. Consequently, any exchanges protected by these keys are vulnerable. In particular, the adversary learns: identities of both peers, the SAs and SPIs of the Child SA created during the authentication process, the AUTH payload, the identity of the PPK. Of course, this means that the adversary can forge these values also.

The keys that are created for Child SAs are unknown to the adversary. This includes the initial Child SA created during the AUTH process. Similarly, all rekeyed values are secret. This is because the keys for Child SAs and Rekeying involve `SK_d` and so involve the PPK.

5.2. **Authentication.** In the vocabulary of the authentication properties that we examine, we consider a section of a protocol to be 'complete' when the initiator and responder have agreed a set of keys to be associated to a Security Association. So for example, the minimal IKE protocol (that terminates after the AUTH payload has been mutually accepted) is complete once a set of keys has been agreed for the first set of SA protocols that were negotiated; rekeying is complete once a replacement set of keys has been negotiated.

The results are captured in Table 1.

TABLE 1. Authentication Results for IKEv2

	IKE SA	**IKE Initial Child**	**Rekey SA**	**Create Child SA**	**Rekey Child SA**
Aliveness I	✓	✓	✗	✗	✗
Aliveness R	✓	✓	✗	✗	✗
Weak Agreement I	✓	✓	✗	✗	✗
Weak Agreement R	✓	✓	✗	✗	✗
Non-injective agreement I	✓	partial	✗	✗	✗
Non-injective agreement R	✓	partial	✗	✗	✗
Injective agreement I	✓	partial	✗	✗	✗
Injective agreement R	✓	partial	✗	✗	✗

5.2.1. *IKE SA.* The first IKE SA is created with full authentication properties. This is under the assumption that Diffie-Hellman is insecure (due to a quantum computer) and that the shared secret (PPK) is known only to the initiator and responder. The shared secret that is used in signing the octets of each party ensures that messages cannot be forged by the adversary.

5.2.2. *IKE Initial Child.* The initial child that is created within the IKE SA exchange has only partial authentication properties. This is because there is nothing binding the SPIs in the header to the content of the exchange: the adversary is able to replay an exchange under a different header. This essentially represents a denial of service attack since a different header means that the identity of the exchange will not correspond to the set of keys required to process the exchange.

If we exclude the adversary's ability to manipulate the SPIs, the initial child then does conform to full authentication in the sense of Lowe.

5.2.3. *Rekey SA, Create Child SA and Rekey Child SA.* The subprotocols Rekey SA, Create Child SA and Rekey Child SA fail the weakest authentication property of aliveness, and thus all subsequent stronger properties. This is due to the fact that these exchanges are secured with the keys $SK_{ei}, SK_{er}, SK_{ai}, SK_{ar}$, all of which are known to the adversary. Thus, the adversary is able to imitate either party during these exchanges. Consequently, it may be the case that e.g. the initiator rekeys but the responder does not. Similarly, the initiator could create a Child SA with the adversary; however, the adversary will not know the relevant keys (since the keys remain secure due to the involvement of the PPK) – hence in this example, the initiator creates an unusable Child SA.

5.3. **Rekey before creating a Child SA.** The documentation of RFC 8784 acknowledges that the encryption and authentication keys created in the initial IKE SA (namely the keys `SK_ei, SK_er, SK_ai, SK_ar`) are known to an adversary who has access to a quantum computer; hence the ensuing negotiated IPsec SA information (including identities of the peers, configuration parameters, Traffic Selectors) that should be protected with these keys is not secret. The RFC imposes the requirement:

> "Deployments that treat this information as sensitive or that send other sensitive data (like cryptographic keys) over IKE SAs MUST rekey the IKE SA before the sensitive information is sent to ensure this information is protected by the PPK."

The RFC suggests creating a childless IKE SA (as specified in RFC 6023 [13]) and immediately rekeying before creating subsequent children with which to exchange information. Although the RFC does not consider the authentication properties of this amendment, we are able to demonstrate that the full authentication properties of Lowe are recovered. That is, assuming that the initiator and responder have successfully created an IKE SA and have rekeyed it (so that both the initiator and responder have a matching set of rekeyed secrets), that a subsequent Child IKE SA has a set of keys not known to the adversary and an adversary cannot deceive either party into creating a Child IKE SA that is not shared with the intended party. Note that the failure of the aliveness property of the Rekey SA aspect of the protocol means that it remains possible for the adversary to make e.g. the initiator rekey without the responder also rekeying; recall though that the adversary will not know these rekeyed secrets.

We conclude that the easy mitigation of rekeying before creating Child IKE SAs provides additional security benefits.

6. Conclusions

RFC8784 offers an lightweight quantum-safe solution to IKEv2 compatible with current implementations. Secrecy of keys is preserved when using a pre-positioned key. The first IKE SA upholds the strongest notion of authentication; rekeying SAs and creating Child SAs fail the weakest authentication properties under the default implementation of the RFC. Full authentication properties can be recovered under careful implementation, namely by rekeying before creating Child IKE SAs. Furthermore, we suggest that the RFC be updated so that rekeying before creating Child IKE SAs be the default option.

A draft RFC [17] proposes an alternative approach for mixing pre-shared keys in IKEv2. A future work will use formal verification techniques to evaluate the security claims of this new approach.

Acknowledgements

The author thanks Roberta Faux, George Galvin and Daniel Shiu for important and useful discussions.

References

[1] Armando A., D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Hankes Drielsma et al. *The AVISPA tool for the automated validation of internet security protocols and applications.* In International conference on computer aided verification, pp. 281-285. 2005.

[2] Blanchet B. and V. Cheval. ProVerif: Cryptographic protocol verifier in the formal model. `https://bblanche.gitlabpages.inria.fr/proverif/` Accessed: August 2022.

[3] Cremer, C. *Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2.* In European Symposium on Research in Computer Security, pp. 315-334. 2011.

[4] Cremers C, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe. *A comprehensive symbolic analysis of TLS 1.3."* In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1773-1788. 2017.

[5] Dolev D. and A. Yao. *On the security of public key protocols* . IEEE Transactions on Information Theory 29(12), pp. 198-208. 1983.

[6] Fluhrer S., P. Kampanakis, D. McGrew and V. Smyslov. "Mixing preshared keys in the Internet Key Exchange protocol version 2 (IKEv2) for post-quantum security." In IETF RFC 8784. 2020.

[7] Gazdag S., S. Grundner-Culemann, T. Guggemos, T. Heider, and D. Loebenberger. *A formal analysis of IKEv2's post-quantum extension.* In Annual Computer Security Applications Conference, pp. 91-105. 2021.

[8] Kaufman C., P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. *Internet Key Exchange protocol version 2 (IKEv2).* No. rfc7296. 2014.

[9] Lowe G.. *A hierarchy of authentication specifications.* In Proceedings 10th computer security foundations workshop, pp. 31-43. IEEE, 1997.

[10] Ninet T., A. Legay, R. Maillard, L. Traonouez, and O. Zendra. *Model checking the ikev2 protocol using spin.* In 2019 17th International Conference on Privacy, Security and Trust (PST), pp. 1-7. IEEE, 2019.

[11] Ninet T., A. Legay, R. Maillard, L. Traonouez, and O. Zendra. *The deviation attack: A novel denial-of-service attack against ikev2.* In 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 66-74. IEEE, 2019.

[12] Ninet T. *Formal verification of the Internet Key Exchange (IKEv2) security protocol.* PhD diss., Université Rennes 1, 2020.

[13] Nir, Y., H. Tschofenig, H. Deng and R. Singh *A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)* (No. rfc6023) (2010).

[14] Shor, P. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM review 41, no. 2 (1999): 303-332.

[15] The SPIN `https://spinroot.com/spin/whatispin.html` Current Version 6.5.1: July 2020. Accessed: August 2022.

[16] Smyslov, V. *Intermediate Exchange in the IKEv2 Protocol.* Draft version 3. `https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-intermediate-03`. 2019.

[17] Smyslov, V. Alternative Approach for Mixing Preshared Keys in IKEv2 for Post-quantum Security. Draft version. `https://datatracker.ietf.org/doc/draft-smyslov-ipsecme-ikev2-qr-alt/`. 2023.

[18] Stevens, S. *Github Code Repository: Formal verification of the post-quantum security properties of IKEv2 PPK (RFC 8784) using the Tamarin Prover [Computer software].* 2023.

[19] The Tamarin Team. 2022. Tamarin-Prover Manual. `https://tamarin-prover.github.` Current Version 1.6.1: August 2021. Accessed: August 2022.

[20] Tjhai, C. M. Tomlinson, G. Bartlett, S. Fluhrer, D. Van Geest, O. Garcia-Morchon and V. Smyslov. *Multiple Key Exchanges in IKEv2* Draft version 0. `https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-multiple-ke-00`. 2020.