INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Towards a Verifiably Secure Quantum-Resistant Key Exchange in IKEv2

Tobias Heider

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Towards a Verifiably Secure Quantum-Resistant Key Exchange in IKEv2

Tobias Heider

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Sophia Grundner-Culemann Tobias Guggemos Stefan-Lukas Gazdag (genua GmbH)
Abgabetermin:	28. Oktober 2019

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 7. Juli 2077

. . . .

(Unterschrift des Kandidaten)

Abstract

Recent breakthroughs in the field of quantum computing have sparked fears that the cryptographic methods we rely on every day may be broken in the near future. Even worse, encrypted network communication protocols like the IPsec suite rely on an asymmetric key exchange to derive a set of symmetric keys used to encrypt network traffic. An attacker storing recordings of the key exchange and the following communication can break the key exchange, extract the symmetric keys and compromise the clear text data once a powerful enough quantum computer is available in the future .

The goal of this work is to make the IKEv2 protocol and thus the whole IPsec communication quantum-resistant, using novel cryptographic methods collected in the NIST post-quantum standardization project, without introducing new security weaknesses. The challenge lies in the constraints of the new cryptographic schemes, such as enormous key sizes, which the IKEv2 protocol was not designed for.

A thorough analysis of the quantum-resistant key exchange methods in regards to their constraints and an analysis of the IKEv2 protocol in terms of its limitations and security properties gives a clear insight of the changes required to support the new methods. Because the security of most methods is unclear, the protocol design should not repeat the mistake to favor a single cryptographic method. Instead, all key exchange schemes should be interchangeable and even combinable in a hybrid key exchange. To ensure the security of the protocol, this work employs a formal analysis driven protocol design approach similar to TLS 1.3.

The resulting protocol, named PQ-IKEv2, is the result of a comparison of previous work on the subject and new original ideas. It enables a hybrid key exchange with all NIST post-quantum key exchange contestants. To assert its security properties, it is subjected analysis using the Tamarin prover. A preceding analysis of IKEv2 confirms the results of previous security analysis and is then extended to show that the new PQ-IKEv2 protocol provides the same security against a more powerful quantum computer attacker. A benchmark of a PQ-IKEv2 reference implementation proofs to be usable in real world settings with an added latency of merely 10% for a hybrid ECDH and post-quantum exchange.

Contents

1 Introduction

Quantum computing is a field encompassing physics, engineering and computer science to develop a new type of computer processor which utilizes quantum mechanical phenomena like superposition and entanglement. The resulting “quantum computer” is theoretically able to solve certain problems that have high computational complexity much more efficiently than it is possible with today’s computers (in the following also referred to as *classical computers*).

While on the one hand this opens many doors, on the other hand quantum computers present a threat to fields that require computational complex problems. An example of such disciplines is the field of cryptography which relies on particular operations, like decryption of a cipher without possession of a specific key, to be unsolvable using a classical computer. In 1994 Peter Shor at Bell Labs discovered a quantum computing algorithm that can solve integer factorization and the discrete logarithm problem in polynomial time, given a fully functioning quantum computer [Sho99] leveraging a high enough number of *qubits* (the quantum computer equivalent of bits). This algorithm, often simply called Shor’s algorithm, can be utilized to effectively make all public key cryptosystems based on those two problems useless. Unfortunately this applies to almost all public key crypto systems (PKCS) effectively in use today. As a result, an attacker in possession of a powerful enough quantum computer has the ability to compromise any of today’s cryptographically protected network communication.

Since Shor’s discovery, the field of quantum algorithms has attracted the attention of the research community and numerous algorithms have been found to provide theoretical exponential speedup for problems in physics and mathematics utilizing the technique which also enables Shor’s Algorithm, called *Quantum Fourier Transform*. The total number of exponential speedup algorithms however is still very limited. Another prominent quantum algorithm found by Lov Grover offers a quadratic speedup for unstructured search problems [Gro96]. While the speedup is less than Shor’s exponential one, Grover’s algorithm comes with the advantage that it can be applied to a broad range of problems. It has been shown that is can be used to find a 128-bit AES key in only 2^{64} iterations instead of 2^{128} with a classical computer, or a 256 bit key in only 2^{128} iterations instead of 2^{256}. While it does not break cryptographic systems like Shor’s algorithm, it raises the required size of both symmetric and asymmetric keys in cryptography. Table 1.2 summarizes the impact of different quantum algorithms and possible mitigations for commonly used classical encryption, signature and hashing schemes.

All explained attacks remain theoretical as there is no quantum computer existing today that is able to execute any of these algorithms with values in the order of magnitude that are used in cryptography. The highest number factored with Shor’s algorithm as of yet has been the number 21 with a technique described in [MLLL^{+}12], which can easily be done with a non-quantum computer today. It is however unclear when quantum computers will be built that can outperform classical computers with these algorithms. Multiple big technology

Category	Cryptographic Scheme	Quantum Algorithm	Mitigation
Symmetric Encryption	AES	Grover	Double key size
Hashing	SHA2		Double output size
Public Key Encryption	RSA		
	ECIES		
Public Key Signatures	DSA	Shor	None
	ECDSA		
Key Exchange	Diffie-Hellman		
	ECDH		

Table 1.2: Impact of quantum algorithms on popular crypgraphic systems

companies including Intel, IBM and Google have reported to have built proof of concept computers with two-figure numbers of qubits. Once a reasonably sized quantum computer is available, all critical communication should employ mitigations against quantum computer attacks. It should also be noted that a possible attack vector is to start recording encrypted confidential communication today and decrypt it with a quantum computer in the future. Contrary to what is sometimes assumed, Diffie-Hellman's forward secrecy property does not protect it from future quantum computer attacks.

To provide information security in a future with quantum computers the scientific discipline of *Post-Quantum Cryptography* has developed. Post-Quantum, or quantum-resistant cryptography aims to develop cryptographic systems that are secure against both quantum and classical attacks. It is not to be confused with the field of quantum cryptography which uses quantum effects itself. Instead post-quantum cryptography is meant to replace broken systems used in today's network protocols and computers and runs on classical computers. In December 2016 the North American Institute for Standards and Technology (NIST) began a standardization process for post-quantum cryptography, requesting public key schemes, signature schemes and key encapsulation methods that can resist quantum computer attacks. Since the beginning of the process, the number of considered schemes was reduced from 69 to 26, with no clear winner in sight as of today.

2 Background

This section summarizes the scientific background required to understand the later chapters of this thesis. It first outlines the IPsec architecture and its protocols, with a focus on the IKEv2 key exchange, and then presents the state of the art of post-quantum cryptography. Most of the information on IPsec is compiled from the IETF IPsec RFCs, most notably [SK05], [Ken05a], [Ken05b] and the most recent IKEv2 RFC in [KHN$^+$14] as well as all relevant extensions.

2.1 The IKEv2 Key Exchange Protocol

IKEv2 is an IETF protocol standard which allows dynamic key agreement for the IPsec protocol suite. The protocol works in two phases. The first negotiates a secure and authenticated channel between two IKEv2 peers which provides a mean of confidential and authentic communication between the two. The second phase which is negotiated using the protected IKEv2 channel establishes security parameters for encrypted ESP or AH flows and the actual IPsec traffic protection. IKEv2 works on two kinds of security associations (SAs), the IKE SA which defines security parameters for IKEv2's communication itself and Child SAs which are used by the IPsec traffic protection protocols ESP and AH.

2.1.1 The IPsec Protocol Suite

Originally introduced as the "Security Architecture for the Internet Protocol" [SK05], IPsec is a secure protocol suite working on the IP level. Fundamentally, this architecture is composed of the two security protocols ESP and AH, the Security Association and Policy concepts as well as the accompanying key management. It is designed as an open IETF standard to provide interoperable secure communication via IPv4 and IPv6. This secure communication is achieved by cryptographic protection of network traffic providing confidentiality, integrity and access control. Because these properties are provided on IP layer, they also apply to all protocol carried over protected IP traffic. A main advantage of IPsec over other encrypted network protocols like TLS for example is that it does not have to be configured on a per application basis but works transparently between hosts and IP networks. Additionally IPsec includes a minimalist firewalling functionality as a form of access control which enforces authentication and integrity protection of IPsec traffic.

Traffic security is provided by the Encapsulating Security Payload (ESP) or the Authentication Header (AH) protocol. Support for ESP is specified as mandatory in the IPsec RFC while AH support is only a recommendation. One reason for this is that AH does not provide any security services that ESP does not provide while ESP additionally offers traffic confidentiality

```
                 BEFORE APPLYING ESP
            ---------------------------------------
      IPv6  |             | ext hdrs |     |      |
            | orig IP hdr |if present| TCP | Data |
            ---------------------------------------

                 AFTER APPLYING ESP
            ---------------------------------------------------------
      IPv6  | orig |hop-by-hop,dest*,|   |dest|   |    | ESP   | ESP|
            |IP hdr|routing,fragment.|ESP|opt*|TCP|Data|Trailer| ICV|
            ---------------------------------------------------------
                                         |<--- encryption ---->|
                                     |<------ integrity ------>|
```

Figure 2.1: ESP tunnel mode, from [Ken05b]

through cryptographic encryption. Because today's computer processors usually provide hardware accelerated cryptography instructions such as the *Intel Advanced Encryption Standard New Instructions*, short *AES-NI*, symmetric encryption as used in ESP has barely any performance impact, which makes it the more commonly used of the two protocols.

2.1.1.1 Modes of Operation

The IPsec protocols AH and ESP support two modes of operation, tunnel mode and transport mode. Tunnel mode is the more commonly used mode, while transport mode is only applied in the special case of end-to-end IPsec communication. Tunnel mode is meant to be used in setups where the IPsec peers function as gateways connecting protected networks. In tunnel mode the entire original IP packet including the IP header is confidentiality and integrity protected. The original packet is encrypted and wrapped in an ESP packet with a fresh IP header that is addressed to the IPsec peer gateway. The structure of an ESP packet using tunnel mode is shown in Figure 2.1.

In transport mode the receiver of the ESP packet is usually different from the receiver of the original IP packet. Upon receive the gateway decrypts the ESP to recover the original IP packet and then simply route the decrypted packet the same way as if it was received in plain text. Transport mode is used for end-to-end connections. The original IP header stays intact and the ESP header is inserted between original header and protected original body. The receiver of a transport mode is always the receiver of the original packet, thus a received packet is decrypted and passed to the upper layer of the network stack. A notable difference to tunnel mode is that in transport mode the original header is not encrypted or integrity protected. An example of the application of an IPv6 packet before and after being encapsulated with ESP in tunnel mode can be seen if Figure 2.1.

```
                BEFORE APPLYING ESP
           ----------------------------
     IPv4  |orig IP hdr  |     |      |
           |(any options)| TCP | Data |
           ----------------------------

                AFTER APPLYING ESP
           -------------------------------------------------
     IPv4  |orig IP hdr  | ESP |     |      |   ESP   | ESP|
           |(any options)| Hdr | TCP | Data | Trailer | ICV|
           -------------------------------------------------
                               |<---- encryption ---->|
                         |<-------- integrity ------->|
```

Figure 2.2: ESP transport mode, from [Ken05b]

2.1.1.2 Security Associations & Security Policy Databases

The IPsec architecture [SK05] specifies three major databases that hold the state required to process incoming and outgoing IP packets in regard to IPsec. The implementation of the databases is not standardized, but their interface is a requirement for every compliant IPsec implementation in order to ensure interoperability between different implementations. The databases are the Secure Policy Database (SPD), the Security Association Database (SAD) and the Peer Authorization Database (PAD).

The Secure Policy Database specifies how network traffic is handled when IPsec is active. More precisely it filters all network traffic by three categories: *DISCARD*, *BYPASS* and *PROTECT*. For the first two categories the SPD acts similarly to a firewall. *DISCARDED* traffic is not allowed to pass through the network interface, *BYPASS* can pass through unmodified. For *PROTECTED* traffic the SPD also holds information on the security protocol to use, allowed cryptographic algorithms and other security protocol options. Traffic that matches a *PROTECTED* rule can only pass the SPD in a protected state, either encapsulated in AH or ESP. This requires an SA with a set of keys and parameters for the protected connection.

The *Peer Authorization Database* (PAD) contains a list of peers that are authorized to communicate via the policies specified in the SPD as well as their associated authentication values such as public keys, pre-shared authentication values or certificates. When an SA management protocol like IKE negotiates a fresh SA the PAD is used to ensure the peer is authenticated and authorized to negotiate said SA.

Active *Security Associations* (SAs) are stored in the *Security Association Database* (SAD). Each SAD entry is associated with a policy in the SPD and holds the parameters for a single negotiated or statically configured SA such as its *Secure Parameter Index* (SPI) as an unique identifier, the negotiated security protocol, keying material for the connection, selected cryptographic algorithms, sequence number counters, lifetime of the SA and other state of the protected connection. When *PROTECTED* traffic passes the IPsec interface of the kernel, the matching SA is looked up in the SAD and then used to encrypt or decrypt the protected network traffic. When no matching SA is available, either the negotiation of a new SA is initiated or the traffic is not allowed to pass through the interface.

IPsec SAs can either be configured statically or managed dynamically by one of the supported key exchange protocols, of which IKEv2 is the most recent. The IKEv2 protocol can not only exchange key material, but also negotiate support for different ciphers and other cryptographic algorithms. The advantage of using a protocol over static configuration lies in the reduced complexity to initially configure and to refresh the keys in many-to-many connection settings as well as in the simplicity of upgrading to newer ciphers or even key lengths.

2.1.2 IKEv2 Exchanges

IKEv2 communication takes place in so-called exchanges. An exchange consists of a request and a corresponding response. They are identified by their exchange types. An IKEv2 message is built up from a header which looks the same for every message as well as a list of payloads. A special payload is the encrypted payload (SK) which in itself contains a confidentiality and integrity protected list of payloads. The types of payloads that may be included in a message are determined by the message's exchange type. The order of payloads in a IKEv2 message is not enforced.

2.1.2.1 Establishing the Initial IKE SA

Two exchanges are required to establish the initial IKE SA: the IKE_SA_INIT and the IKE_AUTH exchange. Usually four messages are exchanged to establish a common IPsec SA, but in some cases more may be required. The IKE_SA_INIT exchange negotiates cipher algorithms, exchanges nonces and exchanges keying material to derive a shared session secret. Then IKE_AUTH authenticates the first pair of messages, exchanges a unique identifier for each peer and usually establishes the first Child SA.

IKE_SA_INIT

The IKE_SA_INIT exchange is the first exchange of each IKEv2 SA. The initiator of the new SA will first generate a new *Secure Parameter Index* (SPI) which is later used as an identifier for the newly initiated SA. It then sends its SPI in the IKE header (HDR), a proposal of supported cryptographic algorithms in the SAi payload, a KEi payload containing a fresh public key (traditionally a Diffie-Hellman public value) and finally a fresh nonce in the Ni payload. Optionally the initiator may append any number of Notify (N) payloads which are used in several IKEv2 extensions to negotiate support for additional features.

The responder receives the initiator's request and first generates its own SPIr. The concatenation SPIi | SPIr is further called the IKE SA's SPI and uniquely identifies the new SA. The responder parses the SA payload containing a list of supported algorithms and returns its own list which must be a subset of the received one and contains one entry per type which the responder wishes to use. Then it sends its own key material in the KE payload and a fresh nonce in the Nr payload. Finally it might reply to the initiator's notifies and optionally include a certification request (CERTREQ) payload.

At this point the responder can use the received and the freshly generated keys and nonces to compute the so called SKEYSEED. It is used to derive a number of shared session secrets that are later used to encrypt and integrity protect further exchanges.

When the initiator receives the IKE_SA_INIT response it first learns the responder's SPIr value and updates its internal SA's SPI. Next it parses the payloads and also generates the SKEYSEED from the common Diffie-Hellman secret and the nonces. At this point both peers have shared encryption and integrity keys and can further communicate in cryptographically protected messages.

IKE_AUTH

There has not been any peer authentication up to this point. Neither of the peers knows who he is talking to at the other end. The problem is solved with the IKE_AUTH exchange that must always follow after the IKE_SA_INIT exchange. The actual IPsec SAs used by the ESP and AH protocols are called Child SAs in IKEv2. Child SAs belong to an IKE SA, which serves as a control channel for establishing and managing IPsec key material. The IKE_AUTH exchange is usually used to negotiate the first ESP or AH Child SA for the IKE SA that is being authenticated, in order to save the round trip time for and additional CREATE_CHILD_SA exchange.

```
Initiator                                                   Responder
HDR(SPI), SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr}  -->
                                   <--  HDR(SPI), SK {IDr, [CERT,] AUTH,
                                            SAr2, TSi, TSr}
```

Figure 2.3: The IKE_AUTH exchange

IKE_AUTH messages are cryptographically protected using keys derived from the SKEYSEED, therefore the first payload in the IKE_AUTH message is always the encrypted and integrity protected SK payload. Inside SK follows an IDi payload which includes a unique identifier for the peer, an optional CERT payload containing a certificate and corresponding trust anchors, the AUTH value which authenticates the initiator and integrity protects the first exchange, an SAi payload which contains the algorithm proposal for an eventual ESP or AH Child SA, and lastly the traffic selectors TSi and TSr which define the traffic that should be protected by the resulting Child SA. The initiator may optionally include an IDr identifier to select the desired peer identity, in case the responder is running multiple identities on the same network address.

The responder asserts its identity in the IDr payload, authenticates its ID and integrity protects the second message and includes all that is needed to negotiate the first Child SA.

The AUTH value consists of a signature over a predefined block of data that differs between initiator and responder. The initiator signs a concatenation of the full first message of the exchange (the IKE_SA_INIT request), the responder's nonce and `prf(SK_pi, IDi_data)` where `SK_pi` is one of the keys derived from the SKEYSEED, `IDi_data` is the content of the IDi payload without the header and `prf` is the pseudorandom function negotiated in IKE_SA_INIT. The responder signs `Message2 | NonceIData | prf(SK_pr, IDr_data)` respectively. This way the AUTH value authenticates the signer with the supplied ID and integrity protects its message of the first exchange, the received nonce, and the claimed identity value. If certificates are exchanged in IKE_AUTH the peers verify the validity of the certificate chain and that the ID values IDi and IDr match the respective name in the certificate.

2.1.2.2 Negotiation of Cryptographic Algorithms

The negotiation of the cryptographic algorithms happens in the IKE_SA_INIT exchange, more specifically in the Security Association payload. The SA payload contains a list of proposals, where each proposal addresses one of the IPsec protocols ESP, AH or IKE. A proposal contains so-called Transforms which are the actual algorithms that are proposed. Each Transform has a Transform Type which can be one the following:

- **ENCR**: Encryption Algorithm
- **PRF**: Pseudorandom Function
- **INTEG**: Intgerity Algorithm
- **D-H**: Diffie-Hellman Group
- **ESN**: Extended Sequence Numbers

Which transform types can or must be included in a proposal depends on the chosen IPsec protocol. An AH proposal can't have an encryption algorithm, while ESP cannot have a pseudorandom function but must have an encryption algorithm. Some combinations can only be modeled by including multiple proposals of the same type. Combined mode encryption and integrity ciphers for example are listed as encryption algorithm, but do not allow an additional integrity algorithm to be specified. In order to propose combined and non-combined algorithms, two proposals have to be sent.

Proposals sent by the initiator are allowed to include multiple transforms of the same type. Two or more transforms having the same type means the peer proposes to use any of the supplied transforms, in other words it is equivalent to a logical `OR` relation between the transforms. A single transform can further have multiple attributes, which model the choice of parameters for the same algorithm, for example the key length of the AES encryption algorithm. A transform may not have multiple attributes of the same type, instead it is required to have two transforms of the same value with the respective attribute set to the desired values.

Figure 2.4 provides an example of a full SA payload with two ESP proposals.

The peer in the example proposes two configurations. The first says *AES-CBC-128 or AES-CBC-192 or AES-CBC-256 as encryption algorithm, and AUTH_HMAC_SHA1_96 or AUTH_AES_XCBC_96 as integrity algorithm and ESN enabled or disabled.* The second is *AES-GCM with a 8 octet ICV with either 128 or 256 bytes key length and ESN enabled or disabled.*

```
SA Payload
   |
   +--- Proposal #1 ( Proto ID = ESP(3), SPI size = 4,
   |     |            7 transforms,      SPI = 0x052357bb )
   |     |
   |     +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     |     +-- Attribute ( Key Length = 128 )
   |     |
   |     +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     |     +-- Attribute ( Key Length = 192 )
   |     |
   |     +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     |     +-- Attribute ( Key Length = 256 )
   |     |
   |     +-- Transform INTEG ( Name = AUTH_HMAC_SHA1_96 )
   |     +-- Transform INTEG ( Name = AUTH_AES_XCBC_96 )
   |     +-- Transform ESN ( Name = ESNs )
   |     +-- Transform ESN ( Name = No ESNs )
   |
   +--- Proposal #2 ( Proto ID = ESP(3), SPI size = 4,
         |            4 transforms,      SPI = 0x35a1d6f2 )
         |
         +-- Transform ENCR ( Name = AES-GCM with a 8 octet ICV )
         |     +-- Attribute ( Key Length = 128 )
         |
         +-- Transform ENCR ( Name = AES-GCM with a 8 octet ICV )
         |     +-- Attribute ( Key Length = 256 )
         |
         +-- Transform ESN ( Name = ESNs )
         +-- Transform ESN ( Name = No ESNs )
```

Figure 2.4: Contents of an initiator SA payload

```
SA Payload
   |
  Proposal #1 ( Proto ID = ESP(3), SPI size = 4,
   |            7 transforms,      SPI = 0x052357ba )
   |
   +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     +-- Attribute ( Key Length = 256 )
   |
   +-- Transform INTEG ( Name = AUTH_HMAC_SHA1_96 )
   +-- Transform ESN ( Name = ESNs )
```

Figure 2.5: Example Responder SA Payload

A responder receiving this payload must now choose a subset of the proposed algorithms, and return them in its own `SA` payload. The payload therefore can only contain a single proposal in which no transform type can be included more than once. In the example above the initiator may choose AES-CBC-256 and AUTH_HMAC_SHA1_96 without extended sequence numbers. A visualization of the response is given in Figure 2.5.

2.1.2.3 Authentication Modes

IKEv2 without extensions defines 3 different modes of authentication: Public Key authentication (PK), Pre-Shared Key Authentication (PSK) and authentication using the Extensible Authentication Protocol (EAP) which itself supports various modes of authentication. The PK authentication mode of IKEv2 is based on the SIGMA protocol family, a key exchange and authentication protocol that was originally developed for IKE(v1). PK authentication mode in its original form supports only RSA and DSA signatures.

The PSK authentication mode, also referred to as MAC, allows the use of a pre-shared value that is then used in an hmac with a fix value and further hashed together with relevant fields of the IKE_SA_INIT message and the shared DH secret. The resulting hmac protects the derived secret, the first exchange and the pre-shared key all at the same time. When PSK is used, the same key is usually used to protect both directions of traffic. EAP mode allows to embed the powerful Extensible Authentication Protocol with all supported authentication modes into the IKEv2 protocol, but can only be used in addition to the PK mode. All IKE authentication modes can be combined asymmetrically, meaning both peers can use different modes of authentication. IKEv2 extensions additionally provide authentication using various ECDSA [FS07] signatures, EAP only authentication [EST10], PAKEs [Kiv11], NULL Authentication [SW15] or a new, more general Digital Signature mode [KS15] that supports a variety of cryptographic schemes and hashes. Because in some cases using a single mechanism of authentication may not be enough, there is also an IKE extension allowing multiple authentication mechanisms to be used in conjunction [KE06].

```
Initiator                                                 Responder

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

Figure 2.6: The CREATE_CHILD_SA exchange

2.1.2.4 Creating Child SAs

After the IKE SA is established as a result of a successful IKE_AUTH exchange a peer may create additional Child SAs, rekey the existing Child SA or the IKE SA using the CREATE_CHILD_SA exchange. In all three cases a new SA is created. Rekeying of an SA is implemented by creating a new SA and deleting the replaced SA. The CREATE_CHILD_SA may be initiated by either of the peers. In case of the CREATE_CHILD_SA exchange "initiator" refers to the initiator of the exchange, not the original initiator of the IKE SA. The exchange consists of a single pair of request/response messages carrying what is needed to create a new ESP, AH or IKE SA.

The payloads are similar to what has been exchanged in IKE_SA_INIT and IKE_AUTH. The SA payload contains proposals for the new Child SA, including encryption algorithm, integrity algorithm, and optionally a Diffie-Hellman group in case a new key exchange should be done for this Child SA. The `Ni` and `Nr` payloads carry a fresh set of nonces used in the key derivation. If a Diffie-Hellman transform was included in the SA payload, the initiator also attaches its new public key. Lastly traffic selectors are exchanged in the `TSi` and `TSr` payloads, to negotiate IP address ranges and ports of traffic that should be handled with this new SA. Once the responder answers with appropriate values, the new keying material is generated using the following formula:

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

`SK_d` comes from the IKE SA, `g^ir` is the optional new shared Diffie-Hellman secret value in case it was negotiated, and omitted in case it was not. Lastly Ni and Nr are the new nonces. A single CREATE_CHILD_SA exchange may result in multiple SAs. For ESP and AH, each SA protects a single direction of traffic, so two new SAs are derived for each selected traffic protection protocol.

2.1.2.5 Rekeying SAs

In order to preserve forward secrecy of connections, IKEv2 allows the rekeying of existing IKE SAs as well as Child SAs after a configured time interval or message count limit. Other reasons to rekey SAs include the depletion of fixed-amount counters in the SA (like the sequence number). Both types of SAs use the CREATE_CHILD_SA exchange for rekeying.

The IKE SA rekey exchange resembles the initial IKE_SA_INIT exchange, except for the exchange ID header field which is CREATE_CHILD_SA. The request contains a new SA payload including proposals for cryptographic algroithms, a fresh nonce in `Ni` and a key share

```
Initiator                                                   Responder

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

Figure 2.7: Rekeying the IKE SA

in `KEi`. In case of IKE SA rekeying the KE payload in the CREATE_CHILD_SA exchange is required, not optional as in the other uses of the exchange. The SA payloads SPI field contains a new initiator SPI for the resulting SA. The responder replies with the accepted proposal and new responder SPI in the `SA` payload, a fresh nonce in `Nr` and its own fresh key share in `KEr`. The message id in the response header must match the message id of the request. The result is a new IKE SA which has all fields (including counters and window size) initialized to 0. This new SA can be used from there on, but the old SA also remains active without explicit deletion.

```
Initiator                                                   Responder

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

Figure 2.8: Rekeying a Child SA

2.1.2.6 Message Fragmentation

IKEv2 Message Fragmentation [Smy14] is an IKEv2 extension that allows to fragment a single big IKEv2 message over multiple UDP datagrams. The goal is to allow IKEv2 to be used in networks where IP fragments cannot pass through because of non-permissive firewalls or network middleware. The motivation was the discovery of several IP fragmentation based attacks which lead to many firewalls filtering IPv6 fragments.

Using Message Fragmentation, IKEv2 messages exceeding a pre-defined size limit are split up into smaller so-called fragment messages. Before it can be used an additional notification of type IKEV2_FRAGMENTATION_SUPPORTED has to be exchanged in the IKE_SA_INIT exchange to negotiate mutual support for the extension. Once both peers have agreed on the support, the sender may split up the following messages.

Only messages containing only the encrypted payload can be fragmented in IKEv2. When the Message Fragmentation was originally introduced the biggest objects sent in IKEv2 were certificates which are usually sent in the IKE_AUTH exchange which is why fragmentation of the IKE_SA_INIT exchange was not a design goal. The content of the original SK payload is treated as a binary blob that is split into pieces smaller than the configured maximum size regardless of boundaries of inner payloads. The resulting chunks are then sent in new messages containing the encrypted fragment payload SKF. It differs from the SK payload only in the header which has two extra fields: *Fragment Number* and *Total Fragments*. The content of SKF is encrypted and authenticated just like the SK payload, so each fragment receives individual protection.

A receiver will wait until it has collected all fragments, then decrypt them, verify them and reassemble them according to their *Fragment Number* header fields. The resulting packet can be parsed as if it had been received as a single message.

Below is an example of fragmenting a message taken from the original RFC document:

```
HDR(MID=n), SK(NextPld=PLD1) {PLD1 ... PLDN}

                         Original Message


HDR(MID=n), SKF(NextPld=PLD1, Frag#=1, TotalFrags=m) {...},
HDR(MID=n), SKF(NextPld=0, Frag#=2, TotalFrags=m) {...},
...
HDR(MID=n), SKF(NextPld=0, Frag#=m, TotalFrags=m) {...}
```

2.2 Quantum-Resistant Key Agreement

Today effectively all key agreement methods used in real world network protocols rely either on the presumed hardness of the discrete logarithm problem or on the hardness of integer factorization. Examples are public key encryption schemes like RSA and ElGamal but also Diffie-Hellman and the more recent elliptic curve Diffie-Hellman (ECDH) key exchange protocols. While the two problems are not equivalent, they both are special cases of the problem of finding a hidden subgroup for finite Abelian groups which can be efficiently solved with Shor's Algorithm.

Quantum-resistant cryptography systems rely on problems that are not known to be solved efficiently by a quantum algorithm. Several problem categories are known that can be used to construct cryptographic encryption, signature and key agreement schemes. Some of them have been known and used for years, others have just been discovered in the recent effort to develop new quantum-resistant cryptographic schemes.

2.2.1 Post-Quantum Cryptography Standardization

As in other fields of cryptography before, the American National Institute of Standards and Technology (NIST) is organizing a standardization project with the goal of finding suitable candidates for real-world use of post-quantum cryptography [NIS19]. Unlike previous standardization processes, this project is not a contest with a single winner. Rather a set of algorithms based on different presumably quantum-resistant problem-classes shall be found. Because of the heterogeneity of these new schemes with regard to key size, speed and security properties there may be multiple "winners" for different applications. Originally, 59 PKE and KE schemes and 23 signature schemes were submitted before the initial deadline in late 2017. Since then the number of considered Post-Quantum schemes was reduced to 17 encryption and key exchange and 9 signature schemes, over the course of 2 selection rounds. There is currently no public information on when and how the process is planned to end and how many more rounds there are going to be.

2.2.2 KEMs and KEXs

In order to understand the NIST submissions it is important to provide some definitions for different nomenclature first, as even academic literature is sometimes ambiguous. As mentioned before, the IKEv2 protocol provides *key agreement*. The property meant here is defined as a mechanism which allows multiple parties to agree on an identical common secret. Cryptographic key agreement methods usually provide key secrecy, so that only the original peers who agreed on the keys have knowledge of it. An important distinction to make between different kinds of key agreement is who can influence the resulting key to what extent.

One solution for key agreement is the key exchange, where both peers' exchange material resulting in a shared secret key. In an ideal key exchange scheme, all peers contribute equally to the final key. A perfect example for a key exchange protocol is the Diffie-Hellman key exchange [DH76]. Both peers exchange exactly the same amount of data (the public share) to derive a common key that is equally influenced by the peers. This is possible because of the commutativity of the modular exponential function used in the original Diffie-Hellman construction.

It has further been shown that the original Diffie-Hellman construction is not the only one allowing for this kind of commutative combination. The same is used in the ECDH protocol, only that it uses elliptic curves instead of multiplicative groups. Of the proposed quantum-resistant protocols in the NIST competition, only few have provided a Diffie-Hellman like key exchange. Those that can be seen as proper key exchange schemes in most cases have to exchange additional parameters in addition to the public share, like generator matrices, seeds or reconciliation helpers. Examples for key exchange schemes in the NIST process are the lattice-based NewHope and Frodo schemes which both exploit a commutative property constructed from the (R-)LWE problem.

Most of the NIST submissions instead is provide a so called key encapsulation mechanism (KEM). KEMs are usually the result of a public key scheme being converted to a key agreement. The most basic transformation is trivial: Given a public key scheme with a public key K_{pub}, a private key K_{priv} and the operations $c = enc(m, K_{pub})$ and $m = dec(c, K_{priv})$ for encryption and decryption, the scheme could look as follows: peer A sends its public key K^A_{pub} to B. B generates a new uniformly random secret key s and calculates $c = enc(s, K^A_{pub})$. c is returned to the sender, who can then decrypt $s = dec(c, K^A_{priv})$. At this point, provided only A knows K^A_{priv}, both A and B know the shared secret s without exposing it to an eventual eavesdropper. A problem in this design is that the key s was actually solely generated by B and A has to trust B to generate a secure key. A better general transformation from a public key encryption scheme was proposed in [Den03], which allows to construct a secure KEM from almost any known cryptographic scheme. An improvement to the construction above is the introduction of a key derivation function (KDF) which, when used with the shared secret s, outputs an algebraic unrelated value s' that has only been seeded with the value generated by B.

The majority of the key agreement methods in the NIST process are actually KEMs. Examples are NTRU-Prime, Crystals-Kyber or SIKE. One reason for the high number of KEMs in the process may be due to NIST's wording for the request which explicitly mentioned KEMs instead of more generally key agreement methods as discussed in [NIS16]. It should be noted that many of the submitted KEM constructions use sophisticated mechanisms to counter the problems of KEMs mentioned before and use cryptographic password derivation functions to derive a key from a common seed resulting from the exchange.

2.2.3 Overview of the Known Methods

The following section will give an overview of problem spaces that have been used to construct cryptographic schemes that can resist quantum computer attacks, a description of some of the more prominent schemes and an assessment of their relevance with regards to the NIST standardization process.

- **Lattice-Based Cryptography:** Lattices are amongst the most promising approaches for quantum-resistant PQC systems that could be used in real world applications. The lattice problems include the shortest vector problem (SVP), the closest vector problem (CVP) and the learning with errors (LWE) problem. In 1998 the lattice-based NTRU PKE system was proposed and has not been broken yet [HPS98]. Lattice-based schemes can be further divided by whether they use structured or unstructured lattices, where the former introduce some structure for a gain of performance and smaller key sizes. An example of a structured-lattice problem is the Ring Learning With Errors (RLWE) problem which resembles the LWE problem but is adapted for polynomial rings over finite fields. This allows RLWE schemes to have smaller keys than schemes based on LWE. On the downside, the RLWE problem's security reduction proves SVP hardness in ideal lattices, not in general lattices. It is believed that solving SVP in ideal lattices is as hard as in regular lattices, but the confidence in structured lattices is lower. Lattice-based systems make up the biggest share of PKE and KEM schemes in the NIST project (9 out of the 17 proposed schemes), as of October '19.
- **Code-Based Cryptography** This approach uses error-correcting codes (ECC) as they are used in information and coding theory, especially the problem of recovering information from a noisy message. In 1978 Robert McEliece published a cryptosystem [McE78] based on the hardness of decoding a general linear error correction code. The public key of the scheme is a random binary Goppa code, a special kind of error correction code disguised as a random linear code. The ciphertext is the plaintext plus random errors. The private key is what is needed to extract the Goppa code from the public key. Because the problem of decoding general linear codes is NP-hard, but decoding binary Goppa codes can be solved efficiently using Patterson's algorithm the construction can be used as a public key encryption scheme. Another notable candidate is the Niederreiter scheme [Nie86] which also applies McEliece's approach to build a crypto system that can be used as a digital signature scheme. Many others have tried to make the original McEliece scheme more efficient by using different kinds of codes, but most of them have turned out to be insecure. Code-based schemes also make up the second biggest group in the NIST PQC standardization project: Currently 7 of the remaining 17 PKE and KEM submissions are based on error correction codes.

- **Supersingular Isogenies** The security of supersingular isogeny (SI) schemes is related to the problem of finding an isogeny mapping between two supersingular elliptic curves with the same number of points. Even though these constructions use elliptic curves, they are not to be confused with elliptic curve cryptography used today. De Feo, Jao and Plut proposed SIDH, a new Diffie-Hellman like key exchange scheme based on supersingular isogenies that provides forward secrecy and has relatively small keys in comparison to other quantum-resistant schemes. The list of presented SI cryptography systems is limited. Only SIDH and its Key Encapsulation Mechanism (KEM) version SIKE have been proposed before the NIST deadline which makes SIKE the only SI scheme to remain in the standardization process. Since then, a novel but promising scheme called CSIDH was proposed in 2018 [CLM+18]. It is built around a similar problem as SIDH and also promises a Diffie-Hellman-like commutative construction, which is a rare property among presumably quantum-resistant schemes.
- **Multivariate Equations** These are a class of systems built around the hardness of solving polynomial equations over finite fields. Most of the proposed systems using this approach have been broken. Prominent examples include the $C*$ crypto system by Matsumoto and Imai [MI88]. Of the few PKE and KA schemes originally proposed in the NIST PQC project, not a single multivariate equation scheme made it into the second round of the standardization process.

3 Related Work

Different previous works have looked at ways to integrate post-quantum cryptography into real world applications. Some have integrated PQC implementations into cryptography software libraries, others replaced old asymmetric schemes in network protocols like TLS with newer lattice or code-based PQKE alternatives. This section gives an overview of the previous work that has been done in integrating PQC schemes into real world software and their results.

3.1 Transport Layer Security

The Transport Layer Security (TLS) protocol, previously named Secure Socket Layer (SSL) is the presumabely most used network encryption protocol today. It is not only used in VPN solutions like OpenVPN, but also in the HTTPS and QUIC protocols which secure most of today's world wide web traffic. According to the Google Transparency Report 90 of the 100 most visited web sites default to the TLS encrypted HTTPS protocol [Goo19]. TLS is implemented in various major crypto libraries such as OpenSSL, Google's OpenSSL fork BoringSSL, LibreSSL and Mozillas NSS.

Many of the popular protocol implementations as well as independent academic researchers have built "proof of concept" implementations of PQC schemes into the TLS protocol:

BCNS: In a 2015 scientific paper [BCNS15] Bos et al. proposed a novel LWE based key agreement scheme which was specifically developed to be used in the TLS protocol. In later mentions the scheme is usually referred to as BCNS, after the names of the authors: Bos, Costello, Naehrig and Stebila. While the paper mentions TLS, the contribution is rather a new general PQC scheme that is motivated by the necessity of PQC alternatives for the TLS protocols. The paper itself does not deal with TLS specific problems in combination with PQC but rather treats the new scheme as a drop in replacement for RSA and DH. The scheme itself can be thought of as a predecessor of many following unstructured lattice schemes, especially Frodo which is one of the NIST candidates entered by the same research group. Douglas Stebila, one of the original BCNS authors is also project leader of the Open Quantum Safe project as well as the author of an IETF internet draft [SFG19] that proposes a hybrid post-quantum key exchange for TLS 1.3 which is not limited to a single PQC scheme.

IETF Work: Various Authors have submitted IETF drafts to introduce different PQC schemes for the TLS handshake. An early draft in 2001 introduced the NTRU scheme [Sin01] as a drop in replacement for RSA in TLS 1.0, but was later abandoned. Later drafts rely on hybrid post-quantum and classical scheme combinations to deal with the low confidence in the newer PQC schemes. A currently proposed draft called "Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS)" [CC19] introduces ECDH-BIKE and ECDH-SIKE, based on hybrid combinations of the NIST entries SIKE and BIKE with the popular elliptic curve Diffie-Hellman key exchange. Contrary to what the names might suggest, SIKE and BIKE are not related. SIKE promises the smallest public keys of all the NIST submissions but is based on the novel supersingular isogenies that are considered immature by some, while BIKE relies on codes which leads to relatively big keys but is mostly considered a well studied problem. Another recent draft by Stebila and Gueron has the goal to provide a scheme agnostic framework to introduce hybrid post-quantum and classical key exchanges in TLS 1.3 [SFG19]. While their approach is very similar to this work, the draft is in a very early stage and does not provide any noteworthy groundwork today. While many solutions have been proposed, none has been adopted and published as an official RFC.

CECPQ1/2: In 2016 Google announced first experiments with post-quantum cryptography in the desktop version of their popular Chrome web browser as well as their own HTTP servers [Goo16]. Their choice of Key Agreement scheme fell on the "NewHope" scheme by Schwabe et al. which they deemed the "most promising" of the ones they investigated on the search for a suitable candidate. The post-quantum scheme is used in combination with the already present and well studied ECDH key exchange. This guarantees that in case the new PQC scheme is broken the connection security degrades to the security of ECDH and specifically Curve25519 which is still regarded secure against classical computers today. The results of their experiments were published in [Lan16]. Because of the additional size of the NewHope exchange, a increase in network latency could be expected which was confirmed by their measurements. According to their report, "[...] the latency of the slowest 5% of connections increased by 20 ms and, for the slowest 1%, by 150 ms". The conclusion of this first experiment was that, even though the new key agreement added measurable latency, a quick deployment of NewHope in TLS 1.2 and 1.3 in case of need would indeed be possible. Two years later in December 2018. Adam Langley et al. announced a new round of experiments with PQC in the Chrome Browser called CECPQ2 [Lan18]. Because of the success of the previous experiments the authors decided to remain with a structured lattice scheme which is used in a hybrid combination with the x25519 elliptic curve. This time the choice fell on the NTRU-HRSS scheme by Schwabe et al. The scheme is one of many structured lattice schemes in the NIST competition for PQC schemes. At the moment of writing Google has not yet published any results and the authors noted that results cannot be expected soon because the experiment depends on the new TLS 1.3 standard which rolls out slowly.

Cloudflare: Similar to Google's CECPQ experiments Cloudflare has published an implementation of the isogeny based SIDH key exchange written in the Go programming language which can be used in Cloudflare's Go TLS library where it can be used in a hybrid key exchange together with x25519. The only official statement for the addition is a blog post [Val17] explaining their setup but no evaluation or internal test results with the new KE have been published.

3.2 IPsec Protocol Suite

There has been considerable previous work in the area of PQKE in the IPsec key exchange protocol IKEv2. The contributions go from popular applications adding experimental support for PQKE schemes, over academic contributions evaluating the feasibility of different schemes, to standardization efforts at the IETF. As of now there is no official IETF RFC document providing quantum-resistance for IPsec, but various solutions have been proposed to make the IKEv2 protocol quantum-safe.

Strongswan: Strongswan is one of the most wide spread implementations of the IKEv2 protocol for Linux. The project implements several non-standard post-quantum key exchange methods, including the lattice based “NTRU” and “NewHope” schemes. They are implemented in the form of IKEv2 DH groups and thus can be used as a replacement for the normal Diffie-Hellman key exchange of the IKEv2 protocol. Because the protocol does not normally allow multiple key exchanges per SA they can not be used in a hybrid way. As they use “private” IDs the quantum-resistant algorithms can only be used between two Strongswan clients. In a special test environment there have also been tests using different PQC key agreement methods provided by “liboqs” [SM16] in a hybrid key exchange similar to what was proposed in an early version of the IETF draft [TTg+19]. The result is a modified IKEv2 protocol that allows the negotiation of an additional hybrid KE in the “IKE_AUX” (now IKE_INTERMEDIATE) exchange in between the IKE_SA_INIT and IKE_AUTH exchanges. The negotiation is done via the optional “Notify” payload. To the time of writing no results of the experiment have been published except for the Strongswan Documentation[1] of the Setup and an introduction talk at the IETF group meeting[2].

Post-Quantum Cryptography for IPsec: In 2014 Ephraim Zimmer published a paper based on his thesis which examined ways to integrate a post-quantum key exchange, specifically the code-based Niederreiter scheme, in the IKEv2 protocol [Zim15]. The result was a Strongswan extension adding the code PQC scheme as a Diffie-Hellman alternative. Code based schemes, and especially the Niederreiter schemes come with the disadvantage of very big keys (>65535 Bytes). As IKEv2 is a UDP-based protocol those keys exceed the maximum message size of the protocol, which is why the author had to resort to the “Hash and URL” method. Instead of the actual key a participant sends an URL and a hash of its public key. The peer then downloads the key from the URL via the HTTP protocol and verifies the downloaded key with the previously received hash. The downside of this method are the requirement for HTTP and the additional protocol complexity due to the side-channel key exchange. The author concluded that code-based PQC schemes come at a cost of big keys and thus worse performance than the classical DH or ECDH key exchanges, but in general a code-based PQC in IKEv2 is practical.

IETF Work: Since mid 2017 there have been efforts to achieve quantum-resistance in the IKEv2 protocol. One approach that has since been adopted by the working group and is waiting to become an official RFC document is “Postquantum Preshared Keys for

[1]https://www.strongswan.org/testing/ikev2-qske/swanctl/rw-qske-l5/index.html
[2]https://datatracker.ietf.org/meeting/104/materials/slides-104-ipsecme-pqc-for-ikev2-in-strongswan-00

IKEv2". The proposed draft adds pre-shared encryption keys that are hashed into the Child SA key derivation hashes. The resulting keys are quantum-resistant as pre-shared keys can not be cryptographically broken by either classical or quantum computer. The obvious downside is that this approach reintroduces one of the fundamental problem that IKE was meant to solve which is manual key management. A second draft named "Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2)" has the goal to develop a well defined set of rules for how IKEv2 can be extended to do multiple key exchanges (where the additional ones will in most cases be PQKE). The content of the draft has changed drastically over 4 revisions and further redesigns can not be ruled out, as the authors try to find the consensus of the working group. At the time of writing the authors propose the use of another IKEv2 extension called IKE_INTERMEDIATE [Smy19] to handle the additional KE. It adds the Intermediate Exchange between IKE_SA_INIT and IKE_AUTH in order to exchange large amounts of data before IKE_AUTH. While the Intermediate Exchange is meant to be used to exchange arbitrary data, it was developed with the PQKE draft as motivation and first use-case. Despite having gone through a number of revisions the IKEv2 PQKE draft is in an early stage and has not yet been adopted by the ipsecme working group.

3.3 Other Implementations

A lot of related work that has been done is neither specific to TLS nor IPsec. Among these are pure implementations of PQKE schemes in popular cryptographic software libraries or other encrypted and authenticated network protocol implementations that implement PQKEs.

OpenQuantumSafe: The open quantum safe project is a research effort lead by D. Stebila an M. Mosca from the university of Waterloo, working on the integration of post-quantum cryptography into network protocols [SM16]. One result of their work is LIBOQS, an open source software library implementing some of the most popular PQC schemes in a single library with a unified API. Additionally the team has integrated liboqs into OpenSSL and OpenSSH to make it usable in a variety of real world projects. Several experimental projects and software forks use liboqs or OQS OpenSSL to demonstrate real world PQC applications.

PQCRYPTO: Another research project in the field of applied PQC coordinated by Technische Universiteit Eindhoven. Besides many of the schemes submitted to the NIST standardization process they have compiled a software library of all NIST code submissions called libpqcrypto [PQC19]. The library unifies the usage of the different schemes under the API of the popular NaCL cryptography library.

4 Requirements for a Secure PQKE Protocol

One of the biggest obstacles in the integration of new quantum-resistant cryptographic schemes into existing network protocols like IKEv2 is the fact that most schemes cannot be used as a drop in replacement for nowadays cryptographic schemes like RSA or ECDH because of constraints in key size, encryption or decryption speed and other special circumstances. On the other side modifications to the protocol come with the danger of introducing new security holes that could be worse than the threat of an adversary leveraging a quantum computer.

The most fundamental requirement for a secure protocol arises from the research question of this work itself. It is the relatively vague requirement to provide security against an attacker in the future that can leverage a quantum computer capable to solve the discrete logarithm and prime factorization problems efficiently using Shor's algorithm. All other requirements for the protocol are in some way derived from this single goal. Many of the requirements arise from constraints in the IKEv2 protocol which was designed around the Diffie-Hellman key exchange, other arise from uncertainties regarding specific properties of the cryptographic post-quantum key agreement schemes presented in Chapter 2.2.

This Chapter starts with a closer look at the existing post-quantum cryptographic landscape and a comparison of the new schemes in regards to properties such as key sizes and encryption speed to find out what changes to the protocol are required to use those new techniques. The goal of this work is to *add* security to the protocol. To make sure there is no loss of security when the protocol is modified, the next section analyzes the existing security properties of the IKEv2 protocol and what is required to preserve those properties. At last, a method to verify the security properties of a network protocol is introduced in the form of assisted formal verification as it was used in the design of several other IETF protocol standards. It will aid in the evaluation of the new protocol by giving a means of comparing the security properties of the modified and the original IKEv2 protocol. In the end a number of clearly defined requirements can be formulated which will guide the design and the following evaluation of a quantum-resistant IKEv2 successor.

4.1 PQKE Requirements

Due to the standardization efforts of NIST and the combined research and development results of several projects like PQCRYPTO and OpenQuantumSafe, a broad range of quantum-resistant key agreement methods utilizing the known quantum-hard problems have been proposed and tested in academic settings. The goal of this work is also the first requirement for a post-quantum protocol: to **protect the IPsec key exchange against a future quantum attack**. The reason none of the published post-quantum schemes has been

Type	Name
Code	BIKE Classic McEliece HQC LAC LEDAcrypt NTS-KEM Rollo RQC
Lattice	CRYSTALS-KYBER FrodoKEM NewHope NTRU NTRU Prime Round5 Saber Three Bears
Isogeny	SIKE

Table 4.1: The NIST round 2 candidates for Quantum-Resistant Key Exchange

officially standardized for existing protocols is not a lack of interest of the responsible parties (for IKEv2 mainly the IETF *ipsecme* working group) in modern cryptographic primitives. In fact many cryptographic schemes have been added to IKEv2 after the original RFC in the from of extensions, for example all supported ECDH curves. Most such extensions define a new static *Transform ID* value identifying the added algorithm, as well as some security considerations and additional checks to be performed by the IKEv2 agents. In the case of ECDH for example, peers have to check that a received point actually lies on the elliptic curve.

The problem with the proposed quantum-resistant schemes is that they have heterogeneous constraints that make it difficult to use them as drop-in replacement for the Diffie-Hellman or ECDH exchange. These constraints include exceptionally big public keys that have to be transmitted over the network, high computational complexity or the requirement to regenerate certain values after each application, all of which have not been considered when designing the network protocols used today.

Because not all schemes that claim quantum resistance can be considered, this work limits its considerations to the schemes that are in the NIST process at this point. More specifically, all key exchange schemes that made it into the second round of the standardization process. A full list of the considered schemes is given in Table 4.1.

4.1.1 Comparison of Schemes

To understand the specific constraints mentioned before it is useful to compare the quantum-resistant schemes in regards to properties relevant for cryptographic systems. The properties

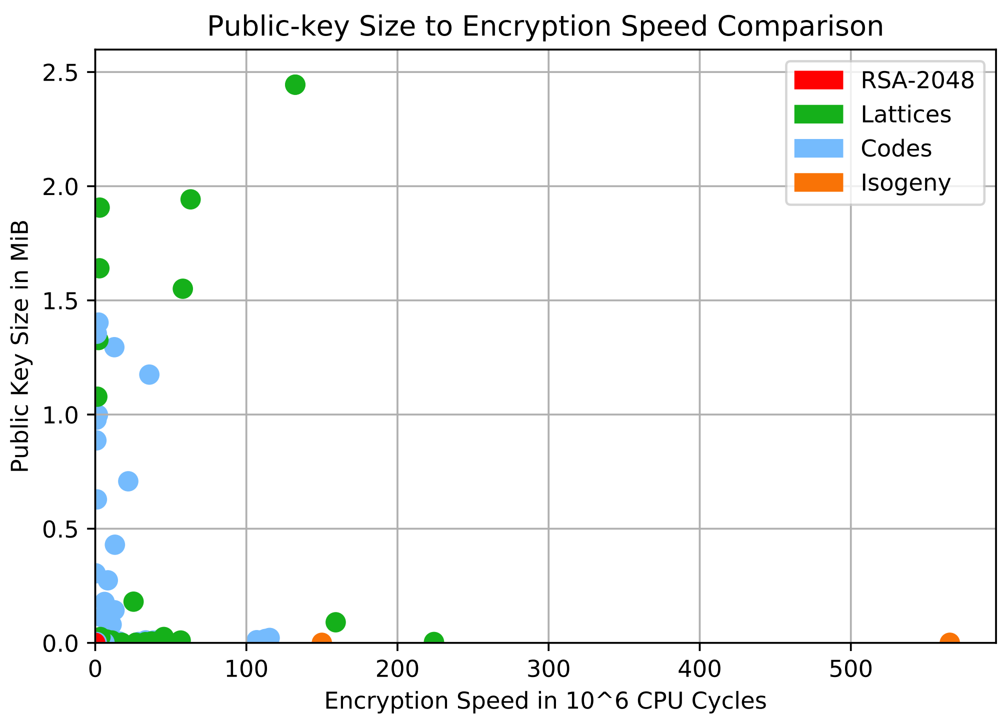

Figure 4.1: The NIST Submissions in Comparison

most suitable for comparison are those that can be reliably measured, such as byte sizes of keys and cipher as well as encryption and decryption speed. The security of the new schemes on the other hand cannot trivially be compared. As most of the proposed schemes have only been published recently and thus have not received in-depth security analysis by a third party.

Because individual security claims cannot be easily verified, a common but informal property used to evaluate a scheme is the *confidence* that exists in the security of the scheme. Older schemes that have been analyzed in more depth and have received more attention from academic reviewers enjoy a higher confidence than newer schemes. This is a relevant factor for quantum-resistant schemes as most are built on hardness assumptions that have been found only recently and that have never been used in the construction of cryptographic systems before. Moreover the NIST process has induced a high number of individual submissions that all happened at the same time, while the number of security researchers in the field is limited. As a result, existing thought to be post-quantum schemes could not receive enough attention in the form academic reviews yet to consider them trustworthy. At the same time even systems that have had public attention and that have received much more analysis in comparison have had security holes that remained undiscovered until many years later, as seen in the *Logjam* attack [ABD$^+$15] that showed how some settings of the Diffie-Hellman key exchange are less secure than assumed before.

Figure 4.1 shows a plot of the NIST PQKE candidates with their problem category denoted by color, the average encryption speed in cycles on the X-axis and the public key size on the Y-axis. The color encoding is explained in the legend. Several patterns can be observed in this graph. The single red dot in the bottom left corner of the diagram is RSA-2048 which was added to set the post-quantum schemes for comparison with a commonly used comparable non-quantum scheme. It becomes clear that the quantum-resistant schemes are all either slower or have bigger keys than RSA. The slowest scheme seems to be around 500 times slower than RSA-2048, the biggest keys are more than 2500 times the size of RSA's. It becomes obvious that, at least with the known methods, resistance against quantum computers comes at a cost.

A notable pattern are the two yellow points at the very bottom of the diagram. They are in the singular isogeny category, more precisely they are two different configurations of SIKE, the only isogeny based scheme in the NIST competition. What makes them special is that they are among the best when it comes to public key size (around 500 Byte), but seem to perform badly in terms of encryption speed compared to both code and lattice based primitives.

For code- and lattice-based schemes it can be observed that some schemes, even though they can almost compete with RSA when it comes to encryption speed, use public keys of immense size compared to the RSA system. At the most extreme keys can grow to over 1 Megabyte of size. In comparison: RSA-2048 has 256 Byte keys, the popular ECDH key exchange needs to send only 32 Byte of public key shares. Public key size is a serious limitation for network protocols. UDP datagrams have a maximum size of 65536 Bytes or 65 Kibibytes and thus can't fit some of the lattice and code-based schemes keys. The IKEv2 protocol has even lower size limitations.

Figure 4.2 shows a comparison of the code-based McEliece scheme with other round 2 candidates. The Y-axis this time shows the public key size in Megabyte. It can be observed that McEliece can have keys up to 1 MB and more in size. In comparison the red line shows the 1280 Byte mark, an important limit for the IKEv2 protocol as this is the minimum MTU in IPv6 and thus the maximum packet size that guarantees that the IPv6 packet is not subject to fragmentation on IP protocol level. IP fragmentation is often seen as a security risk, with the result that network middleware and firewalls block IP fragments from passing through. This constraint makes many of the post-quantum schemes in this comparison unusable in such networks. Not only most code-based constructions but also almost all of the unstructured lattice schemes have public key sizes of more than a Kilobyte. Only structured lattice schemes and SIKE which is based on supersingular isogenies can be used without a workaround in those cases.

Even though those keys seem unpractical compared with what we have used before, there is a reason the schemes are still in the NIST competition (unlike the more than 70 schemes that have dropped out by now). As mentioned before, the *confidence* in the security of the new quantum-resistant schemes is generally low, as most have not been thoroughly been analyzed and field-tested. The McEliece scheme is a rarity because it has been published in 1978 which makes it the oldest scheme in the competition followed by NTRU which was first proposed in 1992. Moreover the McEliece scheme has withstood years of cryptographic analysis without being broken, giving it a fair amount of confidence compared to the PQC competitors. This makes McEliece almost the only choice for high-security applications and the only recommended KEM by the PQCRYPTO project [ABB$^+$15]. On the other hand it is not practical to use in low bandwidth or low memory settings where the newer SIKE may be

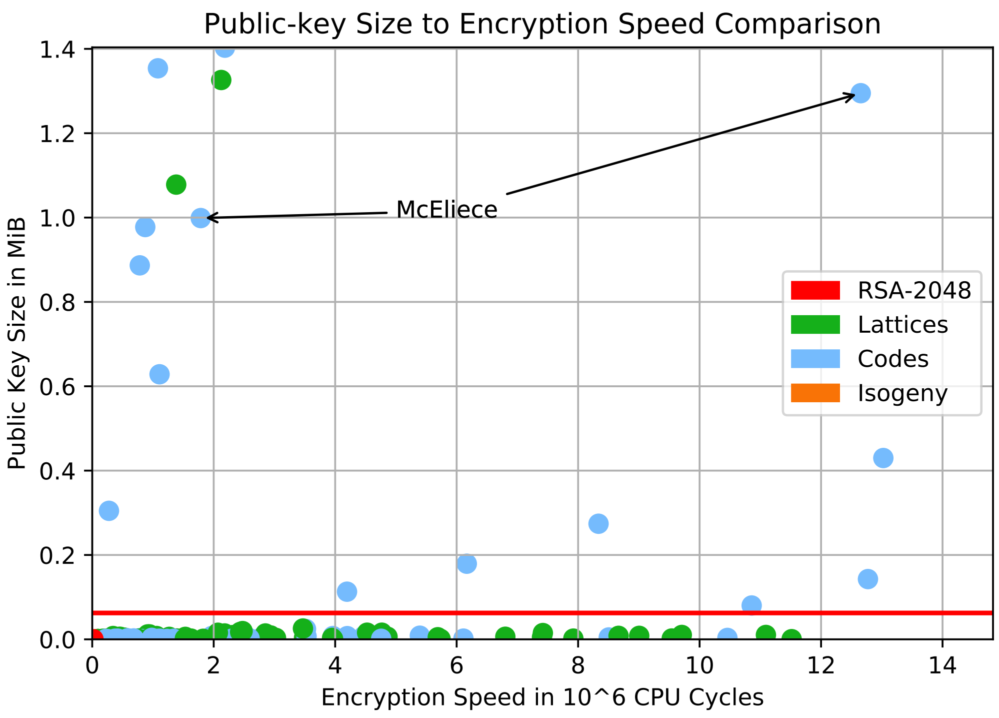

Figure 4.2: McEliece in Comparison

preferred for its small size which again may be too slow for some other settings that require low latency.

The number of key exchange schemes in scope cannot easily be reduced because their probably most important attribute - the provided security - cannot be reliably judged with today's knowledge. All of the compared schemes present a trade-off between speed, communication overhead and security. Consequently, a quantum-resistant key exchange protocol should be designed with respect to the fact that any of the schemes may turn out to be broken, thus it is crucial that it can easily replaced with one of the remaining. This property is called **cryptographic agility** or short crypto-agility. The protocol is responsible to support cryptographic agility by providing facilities to cope with the specific constraints of the different key agreement methods instead of the protocol relying on the properties of a single method. Consequently a second requirement is the **Support for big keys** as they are needed for the McEliece scheme.

4.2 The IKEv2 security model

The IKEv2 standard does not provide a detailed description of the desired security properties of the IKEv2 protocol. Only few properties are explicitly mentioned, including: secure key exchange, forward secrecy [p. 44], identity protection [p. 9], message authentication [p. 9]. For a more de

4.2.1 The IPsec Key Exchange Protocols

The IKEv2 protocol is not the first IPsec key exchange protocol. Many protocols before have tried to solve the same problems and some of them have been the subject of comprehensive security research and analysis. Early IPsec key exchange protocols like the Firefly and the Photuris protocol have relied on the STS key exchange protocol for authenticated key exchange which combines public key encryption with a Diffie-Hellman key exchange.

Other protocols that have emerged in the IETF standardization process for a IPsec key exchange protocol have been SKEME which is similar to Photuris but makes Diffie-Hellman optional and adds fast rekeying without adding much complexity and the Oakley protocol. The STS key exchange that most of those protocols were based on was later shown to have several drastic security flaws. The result was the original IKE protocol which used a signature authentication mode later called the SIGMA protocol [Kra03]. One of the major design decisions with SIGMA was that it provides built-in identity protection, a feature that many of the previous protocols failed to provide. The protocol as it is used in IKE has been analyzed in detail by Krawczyk et al in [CK02].

The history of IPsec key exchange protocols shows that the design of a protocol is not trivial and previous protocols like the original IKE turned out to have major design flaws that were only discovered after publication and adoption as an IETF RFC. The goal of this work is to find a way to make IPsec and especially the IPsec key exchange secure against a future quantum computer attack, which also implies security against today's non-quantum attacks. Because of this, the presented work limits its scope to the IKEv2 protocol and does not consider the design of an entirely new replacement. The IKEv2 protocol has withstood previous cryptographic analysis without major protocol flaws or attacks.

4.2.2 Security Properties

Based on what was learnt from previous protocol's failures, a number of desired security properties for authenticated key exchange protocols can be derived which were used in the design of the IKEv2 protocol. The IKEv2 RFC [KHN$^+$14] itself does not go into detail about what security properties are to be guaranteed by the protocol. In a previous formal analysis of the IKE and IKEv2 protocols [Cre11] came up with the following list of explicitly mentioned security properties: Obtain *authenticated keying material*, Diffie-Hellman security properties as described in Oakley, *mutual authentication* and *perfect forward secrecy.* Authentication is defined informally as two parties agreeing on a their peer identities. The different "strengths" of authentication properties have been explored in detail in the work by Lowe in [Low97]. The description given in [KHN$^+$14] corresponds to Lowe's *weak authentication* property. The original design rationale for the IKEv2 protocol in [Har02] further discusses *identity hiding* against active and passive attackers as one of the original design goals of the protocol. Further information on IKE's security properties can be found in the SIGMA paper by Krawczyk in [Kra03] which formalizes the authenticated key exchange method which was the basis for the IKEv2 public-key authentication. It provides a more precise definition of the authentication and secrecy properties and adds *consistency* to the list of desired properties.

For this work, the desired security properties (compiled from the before mentioned sources) that the IKEv2 protocol and consequently also the quantum-resistant IKEv2 protocol must fulfill in order to provide a secure authenticated key exchange are defined as follows:

Authentication Both parties are able to uniquely identify the other party with which a handshake was performed. The notion from [KHN$^+$14] corresponds to *weak agreement* as defined in [Low97]. Authentication properties can be subdivided into *aliveness* and *agreement*. The *weak agreement* property as it is described in [KHN$^+$14] by definition implies the weaker *aliveness*, meaning every time one party completes a run of the protocol, it implies that the other party has previously been running the protocol. When in addition as a result of the sucessful handshake, the initiator and responder agree on a number of session properties such as session keys, lifetime, and used algorithm, the protocol satisfies *non-injective agreement* or simply *agreement*.

Consistency When two uncorrupted parties have agreed on a matching session, they also have a *consistent* view of the session keys. When the initiator I establishes a session with the responder R, then if R establishes the same session with initiator I then both parties must output the same key K.

Key Secrecy When two honest peers successfully derive a shared session secret, an attacker or passive attacker cannot distinguish the secret from a random string of the same length. In simple terms: no third party can learn any information about the derived secret. Former session secrets should additionally be protected in spite of the compromise of long-term secrets. This property is known as “perfect forward secrecy”.

Identity Hiding As a requirement for the strong authentication property the peers identity has to be communicated some time during the handshake. This identity value should be protected. As the identity is required for authentication, it is not possible to protect both identities against an active attacker as explained in [Kra03]. The KE should protect both identities from a passive attacker and the responder identity from an active attacker.

4.3 Formally Verified Security

Now that it is clear what security properties a quantum-resistant IKEv2 must satisfy, an open problem is how to make sure the protocol actually does so. In the original design of the IPsec and IKE protocols, the design process of the protocols followed a very informal process. Whether a protocol was thought to provide a certain security property was based mostly on the consensus of the IETF participants. Even though they certainly were experts in their respective fields, this design process lead to various major design flaws and security weaknesses in earlier protocol designs which were only discovered in later formal analysis [Cre11] [Mea99].

Over the last decades there have been considerable research achievements in the area of symbolic analysis of security protocols. Formal verification tools have been successfully employed in the development of new protocols to find attacks early or show their absence. Multiple flaws and attacks in established network protocol standards have lead to a rethinking of the classical development process. IETF working groups like TLS have shifted to a “analysis-prior-to-standardization” approach for their recent protocol version 1.3 and were, with the help of academic verification contributions, able to successfully find and eliminate flaws in early design before they could do any harm [CHH$^+$17].

The IKEv2 protocol handles the secure key exchange and the authentication which makes it a single point of failure for the IPsec architecture. While there has been a substantial amount of formal analysis for the IKE (Version 1) protocol, the current IKEv2 protocol has only received limited attention in terms of verification and analysis. The most comprehensive work was done in [Cre11] which found several non-critical violations of strong authentication properties in the IKEv2 protocol.

The modification of the IKEv2 key exchange messages is a drastic change to the protocol design and could have severe consequences for the protocols security properties. A requirement for this work thus must be to **formally analyze the new protocol to ensure the required security properties**. Because no existing IKEv2 analysis has been done with modern verification tooling a second requirement arises: **Before the modified protocol can be verified, a formal model of the original IKEv2 protocol must be built and analyzed for its security properties**. Because the goal is to find a way to integrate post-quantum security into the IKEv2 protocol, the IKEv2 model can then be extended to a post-quantum IKEv2 model while the definition of security properties and basic functionality will not need additional changes, making it easier to compare the two protocols. Previous analysis of IKEv2 as seen in [Cre11] or [CHH$^+$17] can serve as a "ground truth" for the validity of the new model. If the formal model correctly represents the protocol, the analysis should come to a similar result as the previous works.

4.4 Requirements and Methodology

The previous sections have derived various very different requirements for a quantum-resistant IKEv2 protocol. Some of the requirements are very specific, others are more conceptual and can not easily be measured. Before the work on the protocol itself can start, the requirements will be structured and sorted in a way that results in a clear methodology for the design and following evaluation of the new protocol.

As a first requirement stands of course the **Quantum-Resistance**. The obvious way to achieve it is the integration of the known post-quantum key exchange schemes into the protocol. Section 4.1.1 explained that the security of those schemes is uncertain, which is why the favored approach to integrate post-quantum key exchanges is with a hybrid solution which relies on a combination of classical and post quantum exchanges.

It has also become clear that no single new key exchange method should be chosen, instead the protocol should provide **Crypto-Agility** and support a choice of key exchange method. This comes with the caveat of having to deal with the various constraints of the proposed schemes, such as keys that require IP fragmentation and thus break the protocol in certain settings, keys that do not fit into single UDP datagrams and computational complex and slow key exchange schemes.

Because of modifications to the protocols most fundamental structure, the key exchange, it must be made clear that the new protocol does not break the existing **Security Properties** of the IKEv2 protocol. For this requirement to be met, the protocol must provide *authentication*, *consistency*, *key secrecy* and *identity hiding*, just like the original IKEv2 protocol does.

The actual security of a protocol is not trivial to determine. One method that is gaining popularity is computer aided **Formal Verification**, which allows to proof or disproof certain properties of an analyzed protocol in the formal model. The design of a quantum-resistant IKEv2 protocol should consider “verifiability” of the protocol by keeping the complexity of the protocol state machine as low as possible. Before the quantum-resistant protocol can be verified, the original IKEv2 protocol must be modeled and analyzed for the security properties, to act as a basis for comparison with the modified post-quantum IKEv2 protocol.

One last requirement that has not been named before is the **Practicability** of the presented protocol. After all it is unclear whether a protocol doing more than one key exchange, eventually using kilobyte sized keys or slow key exchange methods can realistically be used in real-world settings. This requirement can be evaluated in the form of a proof of concept implementation of the protocol performing several test runs of a *key exchange benchmark* comparing the latency of the post-quantum protocol against the original IKEv2 protocol.

An overview of the collected requirements is given in Table 4.2.

Requirement	*Subgoals*
Quantum-Resistance	Integration of post-quantum schemes
	Hybrid key exchange
Crypto Agility	Support keys problematic because of fragmentation
	Support keys problematic because of UDP size limit
	Support computational complex schemes
Security Properties	Authentication
	Consistency
	Key Secrecy
	Identity Hiding
Formal Verification	Verification friendly protocol design
	Security proof for IKEv2
	Proof to quantum-resistant IKEv2
Practicability	Proof of work
	Benchmarking

Table 4.2: Summary of the requirements analysis

5 Design of a Quantum-Resistant IKEv2 Protocol

The previous Chapter 3 has shown that there have been various attempts to modify existing network protocols - and even the IKEv2 protocol specifically - to support the use of post-quantum key exchange schemes. Most implementation specific proof-of-concept implementations of PQKE in IKEv2 presented solutions that are limited to only one or a selected subset of the considered PQKE schemes. The standardization work in the IETF IPsecme working group on the other hand has evolved to search for a generalized solution to use any not further specified key agreement schemes combined in a so called hybrid key exchange. The proposed solution allows most of the PQKE schemes considered in this work to be used in IKEv2 but fails to address some of the constraints presented by the others.

To design a protocol that can satisfy all the requirements extracted in Chapter 4 some major modifications have to be made to the IKEv2 protocol. Some of the presented requirements have already been solved by related work, in which case their solutions have to be analyzed and then compared. Some of the related work also contains explanations for alternative designs that have been discarded, which will also be considered in the comparison. Other constraints have not been considered in previous work. In this case the presented solutions may be novel and thus have not received any previous discussion or review. Factors that go into the comparison include how well a particular solution resolves the before mentioned constraints and also how **intrusive** the change is to the original IKEv2 protocol design. *Note:* less changes to the protocol mean lower probability that a new protocol vulnerability is introduced and thus resulting in a higher confidence in the security of the presented protocol.

This section goes on to present a detailed description of all modifications to the IKEv2 protocol that solve any of the requirements from Chapter 4. A comprehensive analysis of possible solutions to each of the before mentioned constraints will be presented which results in a well defined quantum-resistant PQ-IKEv2 protocol utilizing the modifications that performed best in each of the compared categories. The following chapters will then evaluate the protocol in regards to the requirements, either using formal methods or experimental proof of concepts and measurements.

5.1 Exchanging Multiple Keys

One of the most notable changes required in the IKEv2 protocol is the introduction of additional key shares that have to be exchanged during IPsec connection establishment. The challenge lies in preserving the security of the protocol as well as satisfying the goals and requirements defined in Chapter 4. Especially the downwards compatibility goal makes this task tricky, as the allowed modifications to the IKE_SA_INIT exchange are limited.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Payload  |C|  RESERVED   |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Diffie-Hellman Group Num    |           RESERVED            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                       Key Exchange Data                       ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.1: Key Exchange payload

In a setting where the initiator supports the new hybrid key exchange extension and the responder does not, the IKE_SA_INIT message sent by the initiator should still be understood as a valid IKE_SA_INIT request in order to get meaningful errors and exception handling. Moreover, the initiator might want to opportunistically offer a hybrid key exchange and downgrade in case it is not supported with the peer. For this to work the peer at the very minimum has to not throw an error on parsing the initiator's request.

5.1.1 Combined Hybrid Exchange

The least intrusive approach for the protocol would be to handle the hybrid exchange outside of the protocol itself, by combining two key exchanges into a single hybrid key exchange which the IKEv2 protocol sees as just another new KE. Given two distinct key exchange mechanism, which both follow the model of exchanging a pair of messages to derive a shared secret, this can be achieved by simply concatenating the exchanged public key shares. Figure X shows the structure of the key exchange payload used in IKEv2.

The hybrid key exchange can be carried out transparently to the transport protocol by encoding it completely in the "Key Exchange Data" field. With two schemes A and B and their public keys PK_A and PK_B, the "Key Exchange Data" would be $PK_A|PK_B$. The data block is treated as a blob in the protocol, the splitting and cryptographic logic can be done in the cryptographic code outside the protocol implementation. The two resulting shared secrets can be combined in a single hash to produce a new single shared secret that can not be found without having the result of both key exchange methods, and consequently fulfills the requirement of a hybrid key exchange. Rekeying can naturally use the exact same procedure, the KE payload data in the CREATE_CHILD_SA exchange works just the same as in the IKE_SA_INIT. Moreover this method does not add any more overhead than the added key material, no extra round trip is required.

One unresolved problem is that the KE payload is matched to the D-H ID in the proposal with the "Diffie-Hellman Group Num" field in the KE header. Unless the combination of KEs can be addressed with a single Transform ID (more on those in Section &negoti), additional reserved space will have to be used. Even then there is only one RESERVED header field

```
Initiator                                                    Responder

HDR(SPI), SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr, KEi2, Ni2 }  -->
                                       <--  HDR(SPI), SK {IDr, [CERT,] AUTH,
                                                SAr2, TSi, TSr, KEi2, Nr2 }
```

Figure 5.2: Additional KE in the IKE_AUTH exchange

left which would allow only one additional ID to be added. If this is not enough, the IDs could be included in the binary blob which would lead to a drastically more complex parser logic in the IKEv2 client and break the assumption that the protocol implementation can treat the data field as a blob. Other notable problems that might arise is that the key shares share a single 16 bit size field in the header which may be a problem for big keys. Even worse size constraints arise in settings where IP fragmentation is not an option. In the case of the initial IKE SA negotiation the first KE payload is sent in the IKE_SA_INIT exchange. IKEv2 Message Fragmentation can only be used on the SK payload which is first used in the IKE_AUTH exchange, and thus does not help in this case. In the IKEv2 specification it says:

"All IKEv2 implementations MUST be able to send, receive, and process IKE messages that are up to 1280 octets long, and they SHOULD be able to send, receive, and process messages that are up to 3000 octets long." - [KHN+14, p. 24]

Considering we are dealing with keys of a **minimum** size of 600 Byte and sizes of more than one Megabyte, this solution can not be used with many of the known quantum-resistant key exchange schemes known today. As a final remark, the solution can not sufficiently satisfy the requirements derived above and is only good for few hand-picked combinations of KE schemes in which we can have only a low confidence today.

5.1.2 Using the IKE_AUTH exchange

To be able to use the IKEv2 Message Fragmentation to transmit keys bigger than the 1280 Byte minimal IPv6 MTU that IKEv2 requires implementations to support, only exchanges after IKE_SA_INIT are eligible. A seemingly trivial solution would be to put the additional key share in its own payload in the IKE_AUTH exchange (together with fresh nonces) that is encrypted with only the initial key exchange. Once the additional shared secret `g^ir` is derived the SA key gets updated with the following formula:

```
    SKEYSEED = prf(SK_d (old), g^ir (new) | Ni2 | Nr2)
```

The initial key goes into the prf function via the `SK_d` value. This updated key computation is the same as is used in the IKE SA rekeying with a CREATE_CHILD_SA exchange. Advantages of this approach are that it uses operations that have been used and cryptographically analyzed before and it allows implementations to reuse their working code. A visualization of the full IKE_AUTH exchange with additional payloads can be seen in Figure 5.2.

A downside of this approach is that it breaks IKEs identity protection under the assumption that an attacker can break the first key exchange. The IKE_AUTH exchange was meant to be confidentiality protected, using this approach this could no longer be guaranteed. This also makes the order of the used key exchange algorithms way more important. Given a setting with a quantum-resistant algorithm which is new and therefore not well researched, and a classical key exchange that is well established but known to be vulnerable to a quantum computer attack, which should go into the IKE_SA_INIT and which into IKE_AUTH? On the other side the post-quantum pre-shared key proposal for IKEv2 also does not allow the IKE SA to be quantum-resistant but only Child SAs which leaves even more communication unprotected to the quantum computer attacker. The reasoning behind this decision was that once quantum computers that are able to break encryption are reality, they will still be expensive to operate and thus no one will use a quantum computer to only reveal meta data like IKE identities.

5.1.3 After Authentication

When performing the additional key exchange after IKE_SA_INIT, the only way to integrate a second key exchange is to rekey the SA with the new derived shared secret. Another design idea that fully abstains from introducing new logic would be to do exactly this, but using the means that are already there. Initiator and responder initiate the IKE SA using the first key exchange. The IKE_AUTH exchange does not negotiate the first Child SA. Instead as soon as the SA is complete, the initiator starts a rekey of the SA with the CREATE_CHILD_SA exchange, only this time it uses the additional key exchange method. One big advantage of this method is that it is the only one that is not suspicable to DOS attacks because the big public keys are only exchanged when the peers are authenticated.

On the downside this approach needs not 4 but 8 messages exchanged until the first Child SA is initiated. This adds notable latency to each connection initialization. Moreover just as exchanging the key in the IKE_AUTH, this solution does not preserve the identity hiding property and a quantum attacker can unveil all data exchanged until the SA gets rekeyed.

5.1.4 Intermediate Exchange

The only solution allowing the exchange of additional key shares after IKE_SA_INIT without leaking identities and other critical data in the encrypted IKE_AUTH exchange would be to introduce a new exchange in between the two. In order to utilize message fragmentation it would need to at least have an encrypted SK payload which requires *some* key material to be present. Another current draft document in the IPsec working group, called the IKE_INTERMEDIATE exchange is exactly this. A new exchange in between IKE_AUTH and IKE_SA_INIT designed to transport big data payloads that do not fit into the initial exchange but need to be exchanged before the authentication (like big public keys of quantum-resistant key exchange schemes).

Adding a new exchange, especially this early in the connection handshake is not a small intrusion into the trusted IKEv2 state machine as it runs at the risk of introducing major new attack vectors and security flaws. One thing that is severely affected is the calculation

```
Initiator                                                          Responder

HDR, SAi, KEi, Ni, [N] -->

                                                <-- HDR, SAr, KEr, Nr,
                                                        [CERTREQ], [N]

HDR, SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr}  -->
                                                <--  HDR, SK {IDr, [CERT,]
                                                         AUTH, SAr2, TSi, TSr}

                          == IKE SA established ==

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

                            == IKE SA rekeyed ==

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

                         == Child SA established ==
```

Figure 5.3: Rekeying after the IKE_AUTH exchange

of the authentication hash in IKE_AUTH. In standard IKEv2 the IKE_AUTH exchange retroactively authenticates the IKE_SA_INIT exchange because there is no way doing it at the time of the exchange. The same counts for an IKE_INTERMEDIATE exchange. The draft [Smy19] modifies the authentication value in IKE_AUTH to:

```
RealMsg1 | NonceRData | MACedIDForI [| IntAuth]
```

for the initiator, and

```
RealMsg1 | NonceIData | MACedIDForR [| IntAuth]
```

for the responder. `IntAuth` is an authentication hash over both peers' messages in all intermediate exchanges that have taken place, so both authenticate not only their own but also the others' intermediate messages. Another difficulty with introducing a new exchange before IKE_AUTH is the error handling which could introduce a DOS attack vector. The solution is to stick to the same rules as the IKE_SA_INIT, which means that receiving an error should not result in an immediate stop of communication, but as a hint to what might be wrong if the handshake fails even after retries.

With this new exchange a straightforward way to do the hybrid key agreement would be to just add the additional KE and eventual nonces to the new messages and after a successful exchange update the SA keys as described in subsection 5.1.2. The design gets more complicated with more than one additional key exchange. Because the IKE_INTERMEDIATE message can be fragmented, it is possible to send more than one KE payload in a single message (as long as the combined keys are smaller than 65 kb). The way message fragmentation works it is not able to resend lost fragments. Instead, the full message and thus all fragments have to be resent when the peer does not respond within a timeout interval. Consequently even with fragmentation active, for a robust connection it is preferred to keep individual IKE messages smaller than the size limit. The full handshake with the additional exchanges in IKE_INTERMEDIATE messages looks as follows:

Advantages of this approach are the reuse of the already existing rekeying logic and is thought to preserve all of the security goals, including identity hiding. On the downside this approach increases the minimum number of messages exchanged with each key exchange. Moreover as the big shares are exchanged before the authentication, this may introduce a DOS attack where an unauthenticated peer can flood another client with big keys.

5.1.5 Rekeying

The design choices for rekeying are similar to the initial key exchange, only here the initial CREATE_CHILD_SA message can use message fragmentation. Moreover the rekeying does not reauthenticate the peers, the question whether the additional KE should be done before or after authentication is no longer relevant. To keep the protocol simple and coherent, the rekeying will use the same mechanism used in the initial exchange.

The **Combined Hybrid Exchange** solution works transparently with all uses of the KE payload. The protocol is not changed, only the content of the KE payload holds multiple keys.

```
Initiator                                                          Responder
HDR, SAi, KEi1, Ni1, [N] -->
                                                    <-- HDR, SAr, KEr1, Nr1,
                                                            [CERTREQ], [N]
                               = Derive Keys =

HDR, SK {KEi2, Ni2} -->
                                                     <-- HDR, SK {KEr2, Nr2}

                                 = Rekey 1 =

                                     ...

                               = Rekey n - 1 =

HDR, SK {KEin, Nin} -->
                                                     <-- HDR, SK {KErn, Nrn}

                                 = Rekey n =

HDR, SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr}  -->
                                            <--  HDR, SK {IDr, [CERT,] AUTH,
                                                     SAr2, TSi, TSr}
```

Figure 5.4: The IKE_INTERMEDIATE exchanges

```
Initiator                                                        Responder

HDR(CREATE_CHILD_SA), SK {SA, Ni, KEi} -->
                               <--  HDR(CREATE_CHILD_SA), SK {SA, Nr, KEr,
                                        N(ADDITIONAL_KEY_EXCHANGE)(link1)}

HDR(INFORMATIONAL), SK {Ni2, KEi2,
 N(ADDITIONAL_KEY_EXCHANGE)(link1)} -->
                                   <--  HDR(INFORMATIONAL), SK {Nr2, KEr2,
                                        N(ADDITIONAL_KEY_EXCHANGE)(link2)}
```

Figure 5.5: The quantum-resistant rekeying exchange

The **IKE_AUTH**, **After Authentication** and **IKE_INTERMEDIATE** solution are equivalent in the case of rekeying, as there is no reauthentication and thus the only distinction to be made is whether to include the news keys in the CREATE_CHILD_SA exchange or send them in a new message after. Even though message fragmentation is used, there is still no way to resend single message fragments. On the downside a solution that sends each new KE in a new message is not as trivial as it was in the initial exchange, as the IKEv2 window size can now be bigger than 1. Sequential order of messages can no longer be assumed, thus a trick is required to link consecutive rekeying exchanges together. Tjhai et al. propose the use of a new Notify payload called `ADDITIONAL_KEY_EXCHANGE` which carries an arbitrary link value. The responder includes this payload with a new link value in every response sent, the initiator includes the same Notify with the same value in the following request. An example from draft [TTg$^+$19] is shown in Figure 5.5

Different to the initial exchange which uses the IKE_INTERMEDIATE exchange ID, the additional rekeying messages have exchange ID INFORMATIONAL as proposed in [TTg$^+$19]. This introduces ambiguity for the parser, which now has to decide whether a received INFORMATIONAL message is a DELETE, an ADDITIONAL_KEY_EXCHANGE or something completely different. A better solution would use a new ID called for example CHILD_INTERMEDIATE which is used in the case of additional material exchanged with a CREATE_CHILD_SA exchange (just like IKE_INTERMEDIATE does for the IKE_SA_INIT exchange).

5.2 Negotiation of Additional KE

The IKEv2 protocol allows peers to negotiate a common set of cryptographic algorithms to be used in their mutual communication. The reasoning behind this design decision is that the IKEv2 protocol is designed to be able to outlast cryptographic algorithms and best practices. By making algorithms a matter of negotiation, the addition of new algorithms to an implementation is made easy without breaking compatibility with non-updated implementations unless intended. Other than for example TLS, the IKE protocol can negotiate arbitrary combinations of supported integrity, pseudorandom, encryption and key exchange algorithms. The alternative design of pre-configured static “cipher suites” of compatible algorithms was rejected in the initial design. In the normal case this design makes it easier to integrate new algorithms into the protocol, as the only thing needed for the new method to be used is an unique ID that both peers understand (instead of new predefined combinations of algorithms or further configuration).

A detailed description of the employed negotiation method was given in Chapter 2. The main problem with the addition of new key exchanges is that the way the structure works there is no way to specify an `AND` relation between transforms of the same type. Another factor that limits possible solutions is that a solution is desired to be downwards compatible with the existing protocol. Because the negotiation happens in the first exchange of the SA, there is no way to negotiate support for an advanced protocol before this point is reached. Several possible solutions manage to provide this additional functionality while staying compatible with non-extended IKEv2 implementations.

5.2.1 Notify encoding

In order to stay downwards compatible it is important to pay attention to what changes from the standard lead to a termination of the IKE exchange and what are actively accepted. When the protocol was designed it was clear that there will be extensions providing new IKEv2 features, as was in IKE (version 1). To allow the addition of new algorithms or other features several payloads and fields are defined to be ignored by implementations in the original standard. An example of such payloads is the Notify payload:

"Notify payloads with status types MAY be added to any message and MUST be ignored if not recognized." - [KHN+14, p. 101]

As a result, status types can be used to negotiate the support for IKEv2 extensions, as is done in the IKEv2 Message Fragmentation. The initiator includes a Notify with the new status type `IKEV2_FRAGMENTATION_SUPPORTED` in the `IKE_SA_INIT` request. If the sender also supports the extension, it will reply with the same Notify in the `IKE_SA_INIT` response. If it does not, it will simply ignore the notify and the initiator when receiving the reply, will know that it can not use message fragmentation with this peer.

The same mechanism can be used to negotiate new key exchanges. For every supported additional KE method a Notify payload of the new notify status type IKEV2_ADDITONAL_KE with the desired KE scheme encoded in its value could be added to the IKE_SA_INIT request. The responder could then, if it has support for the new methods and thus knows the status type, reply with the one it prefers. The new protocol would be downwards compatible because an unknowing peer would ignore the notify messages and would not return any which is equal to selecting none of the new KEs for the initiator, who could then decide to drop the connection or to proceed with only a classical key exchange.

A similar method was proposed in an early revision of the Hybrid PQKE draft [TTg+19]. In their proposal the negotiation of extra key exchanges took place in a `N(EXTRA_KEY_EXCHANGE_POLICY)` notify. The notify payload contained a new binary structure that encodes the full list of proposed additional key exchanges.

The downside of this approach is that it uses a side-channel in the actual protocol and thus makes the protocol state machine unnecessarily complicated. Using side channels like the Notify for anything other than simple yes or no negotiation has, other than the SA payload, not been tested in any way which allows no confidence in the security of doing so. Moreover it makes the formal analysis of the protocol unnecessarily hard. The same of course counts for the final implementation which would in this case require to provide a second parser for the new proposal format.

5.2.2 Extending the Proposal Substructure

In order to be coherent in the design of the protocol, it is desirable to have all negotiation in one place. To achieve this, the new key exchanges have to be added to the existing SA payload proposal structure, preferably in a way that does not require changes in the structure itself as it is already quite complex. Another thing to note is that by introducing complexity to the structure, the actual size of the payload can grow immensely (Especially when more than one proposal is required). A efficient solution thus should use the existing structures.

The new post-quantum key exchange algorithms are not the first set of new key exchange schemes to be added to IKEv2. The Internet Assigned Numbers Authority (IANA) is responsible to maintain a list of standardized key exchange methods for the IKEv2 protocol. When a new KE scheme is introduced, this scheme is assigned a unique transform ID in the Diffie-Hellman Group category that can then be used in the IKEv2 SA payload with compliant implementations. In order to preserve downwards compatibility, unknown IDs are simply ignored by the responder. The most recent example for such an extension are the Curve25519 and Curve448 elliptic Diffie-Hellman curves which have been introduced in [NJ16].

Adding a new quantum-resistant key exchange can be as easy as adding a single new transform ID. The problem with this approach is that it does not yet satisfy the desire for a hybrid key exchange where a classical scheme is combined with the quantum-resistant scheme to have the security of the classical algorithm as a fallback.

An easy approach to negotiating hybrid combinations without touching anything but the Transform ID field would be to specify combined transforms. This method is somewhat similar to the Chrome experiments mentioned in Chapter 3 where they introduced the *CECPQ1* scheme which is effectively a combination of *newhope* and *x25519*. This approach has been used and tested extensively in previous work and implementations, see Chapter 3.

The possible downside of this approach is that the number of required IDs grows logarithmically with the number of schemes that should be combined. If instead only a limited number of combinations are defined, there is the risk of those combinations getting cryptographically broken without having suitable replacements at hand. Another more historical reason speaking against this is that it introduces what is basically key exchange cipher suites which is something the IPsec working group actively avoided in the design of the original IKEv2 protocol

If combined mode transfers can not be used, the only other option is that each new KE is added with its own unique transform ID. An unresolved problem that remains is that the proposal substructure only allows a logical "OR" relation between transforms of the same type. An "AND" relation can in the current form only exist between transforms of different types. Further there can only be *one* chosen transform for each type.

```
SA Payload
   |
  Proposal #1 ( Proto ID = IKE(1),
   |             4 transforms )
   |
   +-- Transform D-H ( Name = CECPQ1 )
   |
   +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     +-- Attribute ( Key Length = 256 )
   |
   +-- Transform INTEG ( Name = AUTH_HMAC_SHA1_96 )
   +-- Transform PRF ( Name = PRF_HMAC_SHA1 )
```

Figure 5.6: Example Initiator SA Payload for IKE SA using a combined transform ID

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Last Substruc |   RESERVED    |        Transform Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Transform Type |     Slot      |          Transform ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                      Transform Attributes                     ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.7: Modified Transform substructure header

One solution to this problem which has not been discussed in the IETF draft because it is an original idea of this work is the introduction of a new field that is called a “slot”. Each transform type can then have multiple slots, which are addressed by their indices. The rules of the proposal substructure are modified so that the responder accepts one transform of each proposed `Transform Type` ∧ `Slot` combination. A logical “AND” relationship between different transforms of the same type can then easily be expressed by assigning different slots. The slot indices can also be used to denote the desired order of the KE algorithms.

Technically, the slot would be an additional field in the Transform substructure header. The original design luckily has two RESERVED fields in the header, a one octet field after the `Last Substruc` field and another one octet field after the transform type field. Because it makes sense to have the two related fields physically closer in memory, the second of the RESERVED fields makes a good fit for the new slot. The resulting structure can be seen in Figure 5.7. The main advantage of this approach is that it treats all KE methods equally. As a result all Transforms of type D-H can be combined in arbitrary order which also allows the hybrid use of two classical algorithms. From an architectural point of view this solution seems elegant as it separates the two problem domains of post-quantum key exchanges and hybrid key exchanges. It provides a solution to the hybrid key exchange problem which reduces the post-quantum problem to simply introducing new D-H Transform IDs as it would

```
SA Payload
   |
  Proposal #1 ( Proto ID = IKE(1),
   |            4 transforms )
   |
   +-- Transform D-H ( Name = Curve25519, slot = 0 )
   |
   +-- Transform D-H ( Name = sntrup4591761, slot = 1 )
   |
   +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     +-- Attribute ( Key Length = 256 )
   |
   +-- Transform INTEG ( Name = AUTH_HMAC_SHA1_96 )
   +-- Transform PRF ( Name = PRF_HMAC_SHA1 )
```

Figure 5.8: Example Initiator SA Payload for IKE SA using slots

be for any other key exchange.

One possible downside of this design is the ambiguity of the new proposed logic with a non-compatible implementation. As for other cases of unknown values that have been explained previously, the `RESERVED` field has to be ignored by standard compliant implementations. A problem arises when the post-quantum aware initiator sends the above proposal and the non-updated responder sees multiple entries with the same transform type and interprets them as "OR"ed, which is not what the initiator intended if he only wanted to use `sntrup4591761` in a hybrid combination and not by itself. The problem could be solved naively by including a notify payload in `IKE_SA_INIT` that if not returned by the responder lets the initiator know the responder was not able to correctly parse the proposal.

A better solution would be to introduce a new Transform Type called "Key Exchange" or KE in short. This new Transform Type would deprecate the old D-H Type. As a convenient side effect the obsolete name Diffie-Hellman Group which was only true when Diffie-Hellman was the only supported key exchange in IKEv2 and the flat hierarchy of the D-H transform type could also be replaced. The new type would not be allowed to be used in combination with the old D-H type. Legacy implementations, if they stick to the standard, would simply ignore proposals containing the new Transform Type. The relevant section in [KHN$^+$14] reads:

"If the responder receives a proposal that contains a Transform Type it does not understand, or a proposal that is missing a mandatory Transform Type, it MUST consider this proposal unacceptable; however, other proposals in the same SA payload are processed as usual." - [p. 88]

The solution also preserves downwards compatibility as it allows the initiator to include a proposal using the new KE type to specify key exchange schemes and a second proposal containing the classical D-H type. A non-compatible responder will ignore the proposal with the unknown KE type and eventually accept the one using D-H if the algorithms match.

Downsides of this approach are the administrative overhead of introducing a new transform

```
SA Payload
   |
  Proposal #1 ( Proto ID = IKE(1),
   |            4 transforms )
   |
   +-- Transform D-H ( Name = Curve25519 )
   |
   +-- Transform AKE1 ( Name = sntrup4591761 )
   |
   +-- Transform ENCR ( Name = ENCR_AES_CBC )
   |     +-- Attribute ( Key Length = 256 )
   |
   +-- Transform INTEG ( Name = AUTH_HMAC_SHA1_96 )
   +-- Transform PRF ( Name = PRF_HMAC_SHA1 )
```

Figure 5.9: Example Initiator SA Payload for IKE SA using new AKE transform type

type and the necessity to move the existing Diffie-Hellman Group Transform IDs to this new type. Another reason to not prefer this approach is that the additional header field is required and the proposal substructure parsing logic would require drastic changes in existing implementations.

Another solution providing almost the same advantages as the slot solution is the introduction of multiple new transform types for additional key exchanges. The hybrid PQKE draft [TTg+19] proposes the addition of 7 new Transform Types named “Additional Key Exchange 1-7”. Those new Transform types are defined to use the same ID table as the D-H type.

A major advantage over the “slot” proposal is that this design leaves the logic of the structure intact, while it allows to combine KE methods with “AND” by putting them in different transform types. The new Transform Types are all optional. If the initiator wants to do only a single KE only the D-H transform is used and Additional KEs are omitted. If an additional KE is meant to be optional, the initiator simply includes a transform of ID none for the transform type. The order of key exchanges is also well defined: the first key exchange in the IKE_SA_INIT exchange is the original D-H exchange, next are the Additional KE types ordered by number where less means earlier. As in the slots design, this allows arbitrary combinations of key exchanges. In a hybrid post-quantum setting the first will almost always be a classical DH or ECDH and the additional KEs can be quantum-resistant key exchanges, but pure classical or post-quantum settings are imaginable. The only restriction imposed is that no two Additional Key Exchanges or the Diffie-Hellman Group must contain equal KEs (because it does not make sense and the KE payload is matched by type).

Downsides to this approach are that the number of Additional Key Exchange Types is predefined and very limited. In the described configuration 8 consecutive key exchanges could be performed for a single connection. This number seems small in comparison to the 8 bit slots field that would allow the negotiation of 256 additional key exchanges. As each additional key exchange adds data and eventually additional round trips to the connection initiation, the real world usability is not only limited by the negotiation but also by the added latency.

5.3 Support For Big Keys

A problem that has not yet been discussed is support for key exchange schemes with public key sizes too big to fit into the IKEv2 message structure. As the size field in IKEv2 payload headers is only 16 Bit it can at max reach 65536 Byte or 65 KB. Especially with code-based key exchange schemes like McEliece or NTS-KEM the public key size can go up to 1 megabyte and higher. IKEv2 message fragmentation is of no help here, because it keeps the structure of payloads intact and fragments and reassembles the message as a blob, so the receiver still has to parse the payload as one.

A solution proposed by many is the use of the “Hash and URL” method. The technique was originally introduced in IKEv2 to transmit oversized certificates in the IKE_AUTH exchange. Instead of including the actual certificate in the CERT payload, the data field contains only a hash of the certificate and a remote URL pointing to the actual certificate. The client can then use another network transfer protocol like HTTP or FTP to fetch the certificate from the URL and verify its integrity by comparing hashes.

The method however has a number of flaws that make it not a good option for exchanging keys. First it requires additional infrastructure in the form of an additional server to host the certificate as well as the client protocol support to fetch the certificate. To make matters worse, the IPsec protocol suite and with it the IKEv2 protocols common use cases include being at the border of a secure firewalled network bridging to other secure networks. The requirement for HTTP or FTP complicates the network setup because it requires additional firewall rules and exceptions. Another problem is that unauthenticated peers can trick clients to pull large amounts of data from remote servers which is a potential DOS attack vector.

A better solution would be to send the big keys in the actual IKEv2 messages. Looking at the structure of an IKEv2 message including UDP header, it is clear that there are several limiting factors. First, the UDP header has a size field of 16 bit. The solution is simple: message fragmentation can be used to fragment the message over several UDP datagrams. The IKEv2 header, even though it is hierarchically higher than the UDP header has a size field of 32 Bit which allows up to 4 Gigabyte of data in a single message. The IKEv2 message hence is not a limiter. The IKEv2 payload header as mentioned above again has a maximum capacity of 65 KB. The proposed solution is called key fragmentation, or in simple terms: to fragment a single key over several consecutive KE payloads in a single IKEv2 message that is subject to IKEv2 message fragmentation. As no two keys can have the same type, consecutive KE payloads with the same transform ID field are clearly identifiable as smaller parts of a single big key. The only addition is a new requirement for the order of payloads in IKEv2. Those KE payloads belonging to a single key have to be sent in row and in order from first byte to last byte of the key. Figure 5.10 shows an example of the IKE_INTERMEDIATE solution using key fragmentation to send a key of up to 325 KB without using the “Hash and URL” method.

Of course this method is still subject to weaknesses in the IKEv2 message fragmentation, especially the loss of single fragments leads to a full retransmit of all fragments. On the same

```
Initiator                                                  Responder

HDR, SAi, KEi1, Ni1, [N] -->

                                        <-- HDR, SAr, KEr1, Nr1,
                                                 [CERTREQ], [N]

HDR, SK {KEi2.2, KEi2.3 ,KEi2.4 ,KEi2.5 , Ni2} -->

                          <-- HDR, SK {KEr2.2, KEr2.3 ,KEr2.4 ,KEr2.5 Nr2}
```

Figure 5.10: The IKE_INTERMEDIATE exchange with a fragmented KE payload

note, message fragmentation should probably not be used with the same configuration it is used for evading IP fragmentation. Considering an IKEv2 message containing a public key share of 1 MB size. This share could be fragmented over roughly $1MB/64KB \approx 16$ KE payloads. Trivial implementations of message fragmentation are recommended by the RFC to fragment to a threshold of 1280 B in the case of IPv6 or 576 B in the case of IPv4. This would mean in the case of IPv6 $1MB/1280B \approx 781$ message fragments or for IPv4 $1MB/576B \approx 1736$ fragments being sent, in both directions. And in case of a single fragment getting lost, all of those need to be retransmitted. A better solution would be to use message fragmentation with a threshold of 65 KB, amounting to 16 fragment messages in each direction which is a more realistic amount for real world settings.

5.4 Final Design

The previous sections have shown that there are many ways to deal with the problems arising with the integration of post-quantum keys into the IKEv2 protocol, nevertheless for the sake of simplicity and compatibility there will only be a single protocol standard. For problems that have more than one solution a choice will have to be made based on the requirements described in Chapter 4 as well as on the previous work of the IETF ipsecme working group and the consensus of experts in the field of the IPsec protocols.

The first problem arising was the integration of additional key exchanges into the protocol. As the original IKEv2 protocol was designed to only use a single Diffie-Hellman key exchange, most solutions to this problem require major modifications to the IKEv2 handshake.

The least intrusive of the presented methods is the combined key method which concatenates two public key shares and treats them like a single key inside the original KE payload of the IKE_SA_INIT exchange. Whilst this method solves the problem with only minimal changes to the handshake and thus the security model, it is not the preferred solution for several reasons: First, it forces the use of combined Group-IDs, which define a static combination of two key exchange methods in a pre-defined constant which must be included in the KE payload for sender and receiver to know how to interpret the received key material. This goes fundamentally against the IKEv2 design which favors free combination of cryptographic algorithms instead of ciphersuites. Second, this method blows up the initial IKE_SA_INIT message, which does not support IKE fragmentation, and thus reintroduces the formerly solved problem of incompatibility with blocked or impossible IP fragmentation.

Exchanging the additional keys in or after the IKE_AUTH exchange both lead to an exposure of the responder's identity and eventual other sensitive data transmitted in the IKE_AUTH exchange. Whilst they also solve problems that other proposed solutions could not, such as the possibility of a DOS attack with big McEliece keys, the information leak directly violates one of the security requirements found in Section 4.2.2 and thus can not guarantee the desired security properties of the IKEv2 protocol.

The proposed solution is the use of the IKE_INTERMEDIATE exchange and specifically to use one IKE_INTERMEDIATE exchange containing a single Nonce and KE payload per additional key exchange, which allows to use IKEv2 fragmentation for the public keys and leaves only the UDP size constraint. Another advantage of this solution is that the payloads and the DH-Group format need no additional changes. The biggest uncertainty with this approach is whether the additional roundtrips will have a too drastic latency impact limiting the usability of the protocol, which is to be evaluated in a proof of concept test setup later on.

For rekeying, it only makes sense to use a similar mechanism as for the initial handshake. The proposed PQ-IKEv2 protocol uses a new CHILD_INTERMEDIATE exchange type to handle additional KEs after the initial CREATE_CHILD_SA exchange, every additional exchange is sent in an additional pair of CHILD_INTERMEDIATE messages.

For the negotiation of the additional key exchange methods, the preferred solution is to reuse existing data structures instead of hacking a new negotiation mechanism into a special notify payload. The two considered approaches are the statically defined "AKE" transform types on the one hand, and the use of a formerly *reserved* header field as "slot", in the specific case of key exchanges a slot value of n marking the n-th key exchange. The downsides of the static types are a pre-defined limitation for the number of key exchanges and more importantly the fact that it breaks the semantics of transform types, introducing n-different types representing the same concept and allowing the same IDs as the already existing group type. Because consistency in the design of the protocol helps implementers and prevents implementation mistakes which could have security impact, the PQ-IKEv2 will instead use the "slotting" mechanism, which in theory could be extended to the other transform types to allow the negotiation of multiple algorithms of the same type. Naturally once the post-quantum key exchange schemes are added to the transform ID registry, they will be treated just like the existing key exchange mechanisms. The concepts of "post-quantum key exchange" and "hybrid key exchange" are actually implemented completely separately when it comes to the protocol.. New cryptographic schemes added to the protocol can naturally be used with the proposed hybrid construction, making it an ideal protocol design in terms of *cryptographic agility*.

The last obstacle is the requirement to support keys bigger than the maximum UDP datagram size of 64 kilobyte. Of the two presented methods, one is to include only a key hash and a URL in the KE payload and delegate the download of the key to another protocol such as HTTP or FTP, which is more a workaround than a solution. The proposed solution is the use of a special kind of *key fragmentation*. Big keys are split up over several KE payloads sent in the same IKE message, which again can be subject to message fragmentation. Using this method a McEliece key could be sent in 16 KE fragments in a single IKE_INTERMEDIATE exchange.

To summarize, the presented PQ-IKEv2 protocol design allows the integration of post-quantum key exchange schemes in a hybrid key exchange, thus fulfilling the first requirement *Quantum-Resistance*. It also provides *crypto agility* by making it easy to switch and freely combine any of the post-quantum key exchange schemes and eliminates the limitation of keys bigger than the IP fragmentation limit by not sending them in the IKE_SA_INIT exchange and using message fragmentation, as well as the limitation of the maximum UDP size using fragmentation over several KE payloads. The design choices have respected complexity of the protocol in order to make the following security analysis not unnecessarily complex.

6 Formal Verification

The security implications of the different designs have only been informal guesses based on previous experience with key exchange protocols, but whether the protocol can actually provide the promised post-quantum security is uncertain. The requirements analysis presented in Chapter 4 lead to a clear definition of security properties a secure key exchange protocol must provide, namely *Authentication*, *Consistency*, *Key Secrecy* and *Identity Hiding*.

A thorough analysis of the proposed PQ-IKEv2 protocol, which is only a modification of IKEv2, is not possible without first analyzing the security properties for the adopted IKEv2 protocol. An easy solution would have been to reuse previous work, but this approach would have limited the choice of weapon to the ones that have been used before. Instead this work includes a whole new analysis of the IKEv2 handshake, using previous results as reference for the correctness of the found results. The validated IKEv2 model can then be adopted to the PQ-IKEv2 protocol.

The analysis is performed using the "Tamarin" tool which has been successfully used in the development process of the new TLS 1.3 standard. It follows the contributions of Cremers in [Cre11] applying a Dolev-Yao attacker model in a try to find logical vulnerabilities in the protocol under the assumption that cryptography cannot fail. Because the Tamarin prover comes with a specific language and syntax for protocol modeling, the chapter first provides a short introduction for the basics of the Tamarin prover, its model specification language and its automated proving capabilities. After this introduction to the prover the IKEv2 model is introduced and explained in detail before the results of the automated verification are presented. The model is then modified with the protocol changes worked out in Chapter 5 and again tested against the same lemmata as the original protocol (eventually adopted to the new model). Finally the results are presented, which show that the quantum-resistant IKEv2 protocol succeeds to provide the same security properties as the original IKEv2.

Used Hardware. While over the last decade automated analysis has become more efficient than ever, the verification of complex models still can be quite hardware expensive. The analysis of the provided IKEv2 model was enabled by the use of the high performance compute cloud hosted at the Leibniz super-computing center in Garching near Munich. The analysis ran on a virtual Ubuntu Linux machine leveraging 40 CPU cores and 180 GB of RAM, which proved invaluable for some of the computationally harder lemmata and especially in the PQ-IKEv2 model.

6.1 The Tamarin Prover

Tamarin[1] is a formal verification tool designed for security protocols which supports automated unbounded symbolic model checking. It supports a specific modeling language that can be used to specify comprehensive models of complex network protocols and adversaries based on the Dolev-Yao attacker model. Tamarin has built-in support for cryptographic functions used in key exchange and authentication protocols. It comes with Diffie-Hellman key exchange, hash functions as well as symmetric and asymmetric encryption and public key signatures included which saves a lot of work in the specification of network security models. 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.

Tamarin has been used for the verification of other popular key exchange protocols. An early work used Tamarin to verify the Noise key exchange protocol framework which later provided the basis for a formal analysis of the Wireguard key exchange. It was also used heavily during the development of the TLS 1.3 [CHH+17] standard to model and verify individual revisions of the draft and helped finding several unintended states in the protocol that might have been used in eventual exploits. Other protocols modeled in Tamarin include the 5G Authentication [BDH+18] or the PKCS#11 standard [KK16].

6.1.1 Example: Modeling an Encrypted Message

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

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

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 `-->`. As mentioned above Tamarin is based on *multiset rewriting*. The global state, or *multiset* is modeled with so-called facts. The rule "Example" requires two facts to be present in order to be executed: `Fr(~sk)` and `!Message(m)`. *Fr()* is a built-in fact that introduces a *fresh* name, in this case `~sk`, `~` additionally denotes a fresh name. In this case the fact models a newly generated random secret key `sk`. `!Message(m)` is a user provided fact named "Message" that is the result of a previously traversed rule. The "!" in `!Message()` denotes a persistent fact. Normally facts in the premise are consumed by the rule and thus removed from the global state, persistent facts can be consumed arbitrarily often.

The result of the rule is a single fact `Out()`. `Out()` is another special built-in fact which denotes that something is sent over the public channel which the Dolev-Yao attacker can observe. The content of the message is the result of a function `senc(m, ~sk)` which is a good example of Tamarin's support for cryptographic functions. It models the result of

[1]https://tamarin-prover.github.io/

symmetrically encrypting the variable `m` with the newly generated secret key `~sk`. The resulting cipher is then sent over the unsecure channel and available to both attacker and recipient

The receiving side could be modeled with the following rule:

```
rule ExampleRecv:
    [In(cipher), !PSK(sk)]
    --[ Received(sdec(cipher, sk)) ]->
    [Recv(sdec(cipher, sk))]
```

or using a helper variable `plain` for the received plain text:

```
rule ExampleRecv:
    let plain = sdec(cipher, sk)
    in
    [In(cipher), !PSK(sk)]
    --[ Received(plain) ]->
    [Recv(plain)]
```

Variables in `let` blocks are resolved with literal replacement, which makes the two examples above equivalent. The premise of the new rule `ExampleRecv` contains two facts. `In()` is a built-in fact and the receiving counterpart to `Out()`. It denotes that a value is received over the insecure channel. `!PSK(sk)` is the pre-shared secret key that was also used by the sender. The result of the rule is the new fact `Recv(plain)`, which resolves to `Recv(sdec(cipher, sk))`. New in this rule is that in addition to premise and conclusion there is also a so called action fact within the arrow called `Received(plain)`. Action facts are observable in *lemmas*, and can be used to determine if certain actions have happened in a trace. Other than premise and conclusion, they do not change the global set of facts. `Received(plain)` is the action of the peer successfully receiving the plain text message.

Security properties of the protocol are defined in *lemmas* in the form of trace properties. They can be automatically proven or disproven by Tamarin's automatic solver. In order to be able to define the desired properties, the right action facts will have to be defined in 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. It reads: 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`.

6.2 IKEv2 Symbolic Model

The first step in the Tamarin aided analysis of a protocol is the specification of an abstract model of the investigated protocol which can be analyzed with the helper tool. The 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 the protocol analysis' validity it is crucial that the model is as close to the actual protocol definition as possible and models all security-relevant information and events correctly. On the other hand, the resource requirements of the Tamarin verification grow 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. To keep complexity at a minimum, a subset of the IKEv2 protocol was selected and then modeled for this work. The chosen IKEv2 subset corresponds to the minimal IKEv2 specification from [Kiv16] with either shared-secret or simple public key authentication. Advanced authentication modes like EAP or certificate validation are not part of the model.

6.2.1 Adversary Model

In order to argue about the security of a network protocol, it is vital to have a well defined adversary model.

In [CK01] Canetti and Krawczyk introduced a formal framework for the analysis of key-exchange protocols arising from a collection of previous work on specific protocols and generalize the results for all key exchange protocols. A key exchange protocol is ned as *"[. . .] a mechanism by which two parties that communicate over an adversarially-controlled network can generate a common secret key."* In their work, they describe the so called Unauthenticated Links adversarial model (UM), which uses a probabilistic polynomial attacker (PPT) who can exercise full control over the communication link. This includes eavesdropping the link, holding back packets or repeatedly sending packets and changing a packets contents. It also controls when protocol events like initiation take place. Additionally it is given a number of oracles allowing the attacker to obtain secret information stored in the clients. This is to ensure that leaked information from clients leads to the least possible effect on the security of the protocol, even over session boundaries. An example is that leakage of long term keys should have no effects on past sessions (perfect forward secrecy). The attacker can use three kinds of oracles: a session-state reveal, a session-key query and party corruption. Last, the model allows sessions to expire. Expired sessions cannot leak state or keys as they are deleted from the clients memory. No oracle can leak data from an expired session. [CK02] made several changes to the model to specifically adopt it to the IKEv2 protocol, the biggest one being that the original model required the identity of the peer to be clear from the beginning, while in IKEv2 identities are only exchanged in the authentication exchange of the protocol.

While the UM model approach provided promising results, more recent approaches, especially when relying on automated tooling to in the analysis, often apply the more commonly used Dolev-Yao attacker model [DY83]. It introduces an attacker that can intercept, eavesdrop, modify and sythesize any network message, the attacker can be seen as the *message carrier*.

In addition to those, the attacker is often given additional capabilities similar to the attacker in the UM from Canetti and Krawczyk, to analyze the impact of the attacker gaining access to several critical variables in the protocols such as private keys or identities. Reasons to apply the Dolev-Yao model over the UM model are a generally wide acceptance in related work as well as the fact that most automated tools have been designed with to model a Dolev-Yao attacker. An example for a work applying this model to the IKEv2 protocol can be found in [Cre11].

For the analysis of the quantum-resistant IKEv2 protocol, the attacker will have additional capabilities to those it possesses in the classical model. It is assumed that the attacker is able to break the classical key exchange at any time, even after the session has expired, as the private Diffie-Hellman secret can be derived from the public share sent over the attacker controlled channel. Even if the peer deletes the secret key from memory that attacker himself can hold back and decrypt the message. To properly model this with an augmented Dolve-Yao attacker, it is provided with a session-key query oracle allowing it access to the otherwise secret non-quantum-resistant ephemeral key.

6.2.2 Model Specification

The specified Tamarin model draws from previous models of key exchange protocols as described in [DM17] and [CHH+17]. In addition, the model uses of a number of built-in cryptographic features such as the Diffie-Hellman equational theory from [SMCB12], Tamarin's symmetric and asymmetric cryptography modules, as well as signatures and hashes. In addition to those, the model defines a new *hmac*/2 function to model HMAC as described in [KCB97], which is needed for IKEv2 key derivation and authentication.

The protocol is modeled as a state machine. Tamarin rules define allowed state transitions in the model. A comprehensive graph of the IKEv2 handshake's state machine is shown in Figure 6.1. Because the full handshake including all optional exchanges is incredibly complex and features multiple loops, the Tamarin model of this work implements only the non-optional parts of the IKEv2 handshake (corresponding to the non-dashed lines in the graph).

The state at any point in time is defined as the internal state of the initiator and the responder, the last message sent over the public channel and any persistent public knowledge such as public keys. The communication peer's state is further divided in the model in a variant and an invariant portion. The invariant is called the peers *Identity* and it contains all state that the specific peer holds persistently even over the scope of a single session, such as its ID value or its long term authentication key or the keys it trusts. The variant state contains values relevant for a single session, such as SPIs, ephemeral keys, or nonces. They are stored in consumable facts and represent knowledge the peer keeps to its next state. The reason to split the state is that having the invariant state in a persistent fact allows the Tamarin prover to apply more aggressive optimizations to the defined constraint model, and thus solve it more efficiently. An example state transition for the IKE_SA_INIT message of the initiator modeled as a Tamarin rule can be seen in Listing 6.1.

As seen in Listing 6.1 the initiation of the protocol handshake `IKE_SA_INIT_I` requires the state invariant `!Identity()` to be defined for the executing peer. As a result, before any

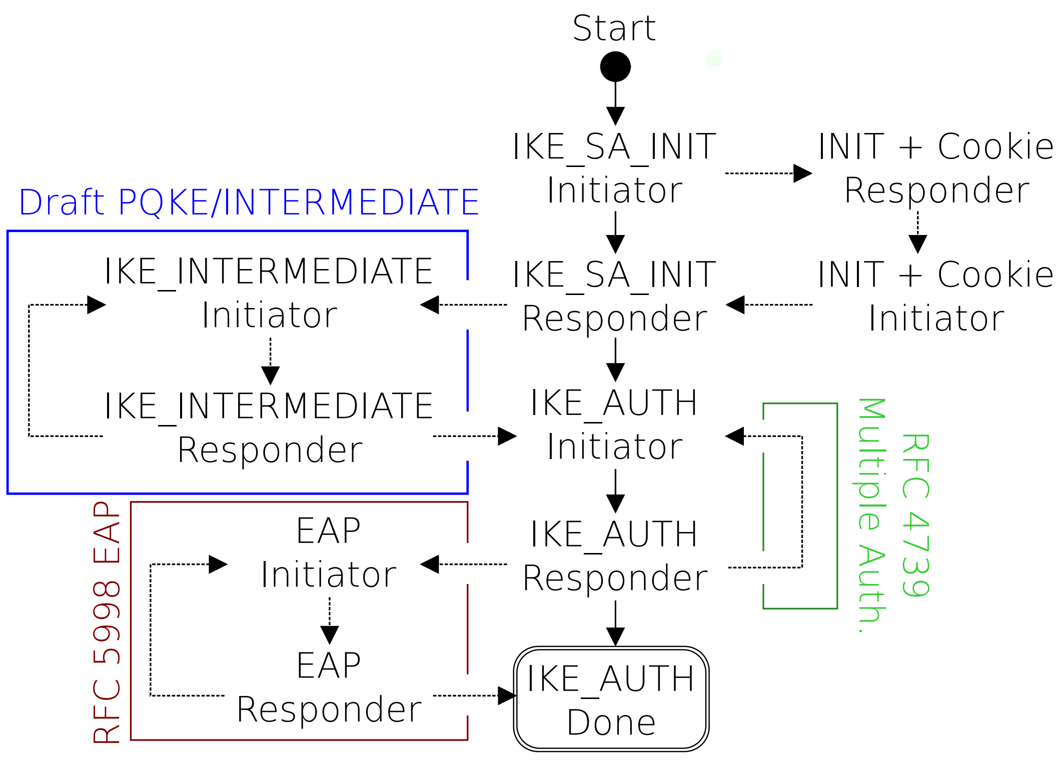

Figure 6.1: The IKEv2 handshake state machine. Dashed lines mark optional paths, extension RFCs and proposed RFC drafts are marked in color. Sources: [KHN+14], [KE06], [EST10]

```
// IKE_SA_INIT Initiator
rule IKE_SA_INIT_I:
let
    epI = 'g'^~eI          // < Initiator DH public share
    msg1 = <~spiI, '1', epI, ~nI>
in
    [ Fr(~nI)              // < Initiator Nonce
    , Fr(~spiI)            // < Initiator SPI
    , Fr(~eI)              // < Initiator DH private share
    , !Identity(idI, pkI, skI, pkR)
    ]
    -->
    [ Out(msg1)
    , StateInitI(idI, ~spiI, ~eI, ~nI, msg1)
    , !DHtoReveal(~eI)
    ]
```

Listing 6.1: IKE_SA_INIT state transition for the IKEv2 initiator role

of the actual IKEv2 protocol communication can take place, the peer identity and keys have to be generated. The identity also models the trust relation with the desired peer, as each identity includes a trusted public key. Key generation and sharing is modeled with the `generate_static` and `add_pk` rules. Generate static is the only rule that takes only fresh `Fr()` facts as an input, and thus has no prerequisites. It generates a new private static key and exposes the corresponding public key over the insecure channel via the built-in `Out()` fact. The `add_pk` fact consumes two persistent static keys, and introduces a new ID value from which it generates a new identity. An IKEv2 session requires at least two unique identities to be defined, one for the initiator and one for the responder.

```
rule generate_static:
        [ Fr(~ltk)]
        --[GenStatic(pk(~ltk))]->
        [ !PrivKey(~ltk)
        , Out(pk(~ltk))
        ]

rule add_pk:
        [ !PrivKey(~ltkA)
        , !PrivKey(~ltkB)
        , Fr(~idA)
        ]
        -->
        [ !Identity(~idA, pk(~ltkA), ~ltkA, pk(~ltkB))
        ]
```

Listing 6.2: Pre-protocol setup rules

Attacker capabilities to reveal certain session state and keying material at different states of the protocol are, like the protocol handshake, realized by transition rules. The attacker is given two different kinds of reveal oracles. They are shown in Listing 6.3. The `reveal_static` rule allows the attacker to compromise a peer's long term key `ltk` by consuming the persistent fact `!PrivKey(ltk)` and exposing the contained key variable over the insecure channel. Similarly a second oracle called `reveal_dh` allows to reveal a peer's ephemeral Diffie-Hellman key share. The preconditioned persistent facts are the result of the generator rules of the exposed values (see Listing 6.2).

```
rule reveal_static:
        [ !PrivKey(ltk) ]
        --[RevLtk(pk(ltk))]->
        [ Out(ltk) ]

rule reveal_dh:
        [ !DHtoReveal(k) ]
        --[RevDH('g'^k)]->
        [ Out(k) ]
```

Listing 6.3: Attacker capabilities as reveal oracles

The validity of the Tamarin model of the IKEv2 protocol, can be verified with trace properties. The `exists-trace` keyword instructs the automated solver to solve for a trace that satisfies a provided property instead of falsifying it. In the case of IKEv2, these traces show that the derivation and confirmation of an IPsec SA are valid in the provided model. In addition it shows that two peers can negotiate an arbitrary number of IKE and IPsec SAs using the same identities, to show that identities and thus ID values as well as static keys can be reused over more than one session. The corresponding lemmas are discussed in more detail in Section 6.2.3.2.

6.2.3 Verified Properties

Section 4.2.2 discussed the claimed security properties of the IKEv2 protocol as well as the results of previous analysis of the protocol. Before the model can be used to analyze security properties of the PQ-IKEv2 protocol it must be able to prove the security properties of an unmodified non-post-quantum IKEv2 protocol. Comparison of the results with previous publications allows assumptions about the correctness of the provided Tamarin model. If the automated proof of the model leads to the same result as previous proofs the model is considered a *valid* representation of the standard and can further be extended to proof the same properties for the PQ-IKEv2 protocol.

The security properties derived in Section 4.2.2 were modeled as Tamarin lemmas over the IKEv2 model. This section introduces the defined lemmas, explain how they verify the security property definitions shown before and presents the results of the analysis of each property for an attacker utilizing the different reveal capabilities.

6.2.3.1 Validity of the Model

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

The first lemma is called `exists_session` and it verifies that the two peers can complete a session that results in an identical shared session key at both peers. The corresponding Tamarin definition is given in Listing 6.4.

```
lemma exists_session: exists-trace
    "Ex I R spi ck #i #j .
        IKeys(I, R, spi, ck) @ i
        & RConfirm(I, R, spi, ck) @ j
        & i < j"
```

Listing 6.4: Session exists, IKEv2

The lemma uses the exists-trace keyword to instruct the automatic solver to find a trace that fulfills the defined property instead of falsifying it.

The second lemma called `exists_two_sessions` goes even further to show that two peers can perform multiple successive handshakes. This property is crucial as some vulnerabilities might require to leak information in one session and reuse it in a second. The corresponding lemma can be found in Listing 6.5

```
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
        & IKeys(I, R, spi2, ck2) @ i2
        & RConfirm(I, R, spi2, ck2) @ j2
        & i2 < j2
        & not (ck=ck2)"
```

Listing 6.5: Two sessions exist, IKEv2

The provided IKEv2 model succeeds to fulfill both lemmas and is consequently ready to be further analyzed for its security properties.

6.2.3.2 Authentication

As described in Chapter 4 the authentication property follows the definitions in [Low97], more specifically it is split up into the three properties *aliveness*, *weak agreement* and *agreement*. For the verification each of those three properties was modeled as a Tamarin lemma.

Aliveness

The weakest of the authentication properties, namely the *aliveness* is formally defined in **Definition 2.1** from [Low97]:

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

The corresponding Tamarin lemma is given in Listing 6.6. Due to Tamarin's language being close to the mathematical logical notation, the lemma is almost a literal translation of the definition.

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

Listing 6.6: Aliveness, IKEv2

No compromised keys When no keys are compromised the IKEv2 protocol achieves *aliveness*.

Compromised ephemeral keys When only ephemeral keys are compromised, the protocol achieves *aliveness*

Compromised static keys When only the initiator's static key is compromised, the protocol achieves aliveness. When the responder's static key is compromised, the attacker can "impersonate" the role of the responder and aliveness is no longer achieved by the protocol.

Compromised ephemeral & static keys When the ephemeral keys and the initiator's static key are compromised, the protocol still achieves aliveness. Same as with compromised static keys, when the responder's static key is compromised *aliveness* is no longer guaranteed as the attacker may take over the role of the responder.

Weak Agreement A stronger definition in the hierarchy of authentication specifications is the *weak agreement* specified in **Definition 2.2** of [Low97]:

> We say that a protocol guarantees to an initiator A weak agreement with another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A.

The corresponding lemma is given in Listing 6.7

```
lemma weak_agreement[use_induction]:
    "All spi A B keymat #i .
        Completed(spi, A, 'initiator', B, keymat) @ #i
        & not (Ex #k . RevLtk(B) @ k)
        & not (Ex #k . RevDH(A) @ k)
        & not (Ex #k . RevDH(B) @ k)
    ==> (Ex spi2 role keymat2 #j . Completed(spi2, B, role, A,
       keymat2) @ #j
        & #j < #i)"
```

Listing 6.7: Weak Agreement, IKEv2

Different than in the *aliveness*, not only does A complete a protocol run with B, but B now also believes to have completed the protocol run with A. The definition of the properties look similar, but the results are different.

No compromised keys When no keys are compromised the IKEv2 protocol achieves *weak agreement*. When a party A believes to have completed a protocol run with B, then B also believes to have completed a run with A before.

Compromised ephemeral keys When the ephemeral key of a single party is compromised, IKEv2 fails to achieve *weak agreement*. A weakness in the digital signature authentication mode due to IDs being optional can result in a situation where one agent successfully completes the handshake while its peer believes to be communicating with someone else. The same weakness was first documented as the result of a previous analysis in [MD$^+$97] and later confirmed in [Cre11]. It has been shown that later communication over the resulting IPsec tunnel provides the even stronger *agreement* property, thus making the weakness a non-exploitable one. It is however notable that the compromise of ephemeral keys can result in a broken authentication.

Compromised static keys When only the initiator's static key is compromised, the protocol achieves *weak agreement*. With the responder's static key compromised an attacker can successfully "impersonate" the other agent, resulting in a session for A while B has not run the protocol with A. Consequently when both keys are compromised, agreement is not guaranteed.

Compromised ephemeral & static keys When the ephemeral and static keys are compromised, the protocol fails to provide *weak agreement* for reasons explained above.

Agreement

The strongest authentication property in the hierarchy is the *injective agreement* or simply *agreement* as defined in **Definition 2.3** of [Low97]:

> We say that a protocol guarantees to an initiator agreement with a responder on a set of data items ds if,whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds, and each such run of corresponds to a unique run of B.

Which translates to the following lemma in Listing 6.8.

```
lemma agreement[use_induction]:
    "All spi A B nonces keymat #j .
        Completed(spi, A, 'initiator', B, keymat) @ #j
        & not (Ex #k . RevLtk(B) @ k)
        & not (Ex #k . RevDH(A) @ k)
        & not (Ex #k . RevDH(B) @ k)
    ==> (Ex spi2 #k .
        Completed(spi2, B, 'responder', A, keymat) @ #k)"
```

Listing 6.8: Agreement, IKEv2

The differences to the *weak agreement* lemma are that B now acts as a responder. Additionally shared variable keymat models the agreement on the set of data items *ds*. In the case of the IKEv2 model upon a successful run of the protocol the two peers agree on the keymat which is derived from the key exchange and the unique nonces .

No compromised keys When no keys are compromised the IKEv2 protocol achieves *agreement*.

Compromised ephemeral keys When the ephemeral key of a single party is compromised, IKEv2 fails to achieve *agreement*. The reason is the same as explained above for the *weak agreement* property. The found weakness is known and is not known to be exploitable in real world settings, see [Cre11] and [MD+97].

Compromised static keys When only the initiator's static key is compromised, the protocol achieves *agreement*. With the responder's static key compromised an attacker can "impersonate" the peer, resulting in a mismatched session between the two.

Compromised ephemeral & static keys When the ephemeral and static keys are compromised, the protocol fails to provide *agreement* for the reasons explained in **Compromised static keys**.

The definition of the *agreement* requires that each of the runs of A corresponds to a unique run of B, which has not been explicitly shown yet. A new lemma called session uniqueness is defined to show that each identical set of key material derived from a successful run of the protocol corresponds to a unique session. The Tamarin definition can be found in Listing 6.9.

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

Listing 6.9: Session Uniqueness, IKEv2

What the lemma specifies is that for two protocol runs to be completed and result in the same `keymat`, they must happen at the same time and thus be identical. Consequently every other successful protocol run must result in a different set of nonces and keys, making each session with the same variable a *unique* session.

6.2.3.3 Consistency

Consistency This property corresponds to the first property from **Definition 1.** in [CK01] which reads:

> "1. Protocol π satisfies the property that if two uncorrupted parties complete matching sessions then they both output the same key;"

The same property was used in [CHH+17] where the corresponding lemma was named `session_key_agreement`. The difficulty in defining this property as a Tamarin lemma lies in finding a suitable definition for the *matching session*. In the case of IKEv2 a session is by definition uniquely identified by its SPI combination, thus a *matching session* is one where both peers see the same SPI pair. The corresponding Tamarin lemma is shown in Listing 6.10

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

Listing 6.10: Consistency, IKEv2

When two agents I and R each complete a run of the protocol, and I acts as the initiator and thinks it talks to R and R acts as responder and thinks it talks to I, then they both agree on a identical set of session variables `keymat`.

No compromised keys When no keys are compromised the IKEv2 protocol achieves *consistency*.

Compromised ephemeral keys When one or both ephemeral keys are compromised, IKEv2 still achieves *consistency*. A compromise of only the ephemeral keys can not change the resulting key and nonces, as they are protected by the authentication signature.

Compromised static keys When only the initiator's static key is compromised, the protocol achieves *consistency*. When the responder static key has been compromised, an attacker can "impersonate" the other party and mount a man-in-the middle attack resulting in the peers having different session keys.

Compromised ephemeral & static keys When both ephemeral keys and the initiator static key are compromised, the protocol achieves *consistency*. When the responder's static key is compromised, the same weakness as in the "compromised static keys" case applies.

6.2.3.4 Key Secrecy

Key Secrecy

As a result of a successful run of the protocol, keys are only known to the initiator and responder. The corresponding lemma states that when a party I has completed the protocol with another party, and it thinks it has during the same unique session agreed on a shared session secret, then the attacker can not also know the secret.

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

Listing 6.11: Key Secrecy, IKEv2

No compromised keys When no keys are compromised, the IKEv2 protocol achieves *key secrecy*. A success completion of the handshake guarantees that only the two agents performing the handshake have knowledge of the shared session key.

Compromised ephemeral keys When either ephemeral key is compromised, a passive attacker can learn the shared session secret as it is derived from the Diffie-Hellman result, which requires the public DH share which is sent in clear and both nonces which are also sent in clear over the insecure channel.

Compromised static keys When the active agent's static key is leaked, the protocol still achieves *key secrecy*. When the peer's static key is leaked, an attacker can mount an impersonation attack and thus learn the shared session secret by performing the initial handshake with the original agent.

Compromised ephemeral & static keys When any ephemeral key is leaked, the protocol does not provide *key secrecy*.

6.2.3.5 Identity Hiding

Identity Hiding

The *Identity hiding* (or *identity protection* differs from the other requirements as it is only an additional security guarantee. As noted in [Kra03] 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.

The Tamarin lemma can be translated as: "When an IKEv2 participant B sends out its identity *id* at *i* to authenticated peer A, the attacker must not be able to learn *id* at any point in time *j*.

```
lemma identity_hiding:
    "All spi A B keymat id #i .
        Completed(spi, B, 'responder', A, keymat) @ #i
        & IdentityLearnt(id) @ #i
        & not (Ex #k . RevLtk(A) @ #k)
        & not (Ex #k . RevDH(A) @ #k)
        & not (Ex #k . RevDH(B) @ #k)
    ==> not (Ex #j .  K(id) @ #j)"
```

Listing 6.12: Identity Hiding, IKEv2

No compromised keys With no compromised keys the IKEv2 protocol achieves *Identity hiding* for the responder.

Compromised ephemeral keys A single compromised ephemeral key leads to a loss of *identity protection*, the attacker can generate the symmetric encryption key, decrypt the authentication response and learn the responder's identity.

Compromised static keys With the responder's static key compromised the attacker can not learn the responder's identity and the protocol achieves *identity hiding*. A compromised initiator static key allows the attacker to perform an impersonation attack which leads to a loss of *identity hiding*.

Compromised ephemeral & static keys As the compromise of a single ephemeral key leads to a loss of *identity protection*, the compromise of both ephemeral and static keys also breaks the property.

6.3 The PQ-IKEv2 Model

After the IKEv2 model has successfully passed the formal analysis confirming the results of previous formal analysis of the protocol, the approach can be adopted to the augmented model of the post-quantum hybrid IKEv2 protocol. Before the result of the post-quantum analysis is presented, the changes made to the model are laid out in detail. It should be noted that the PQ-IKEv2 model is more complex than the original, as it adds an additional exchange and additional attack surface to leak secrets, which makes the automated analysis even more computation-intensive.

6.3.1 Changes to the IKEv2 Model

For the post-quantum hybrid exchange, the original IKEv2 model was extended with the new IKE_INTERMEDIATE exchange. The intermediate exchange can be seen in Figure 6.1 outlined in blue. The additional exchange is implemented in the form of a new transition rule between the IKE_SA_INIT and IKE_AUTH rules. In the formal model, the post-quantum key exchange is also modeled with the built-in Diffie-Hellman module. As the model assumes perfect cryptography, there is no difference between a quantum-resistant and a non-post-quantum key exchange. Instead, the new secret key is leaked with a new fact called `!DHQtoReveal` which can be distinguished from the `!DHtoReveal` fact used for the first key exchange. A post-quantum attacker simply is given a revel transition for the non post-quantum key. The Tamarin notion of the initiator's new rule is presented in Listing 6.13.

```
// IKE_INTERMEDIATE Initiator
rule IKE_INTERMEDIATE_I:
let
        msg2 = <spiI, spiR, 'IKE_SA_INIT', '1', 'r', epR, nR>

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

        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, $R)
        ]->
        [ Out(msgINT)
        , StateIntermI($I, $R, spiI, spiR, nI, nR, ~nI2,
            keymat, msg1, msg2, msgINT, 'g'^eI, epR, ~peI)
        , !DHQtoReveal($I, ~peI)
        ]
```

Listing 6.13: Initiator IKE_INTERMEDIATE rule

As a result of the new exchange, the IKE_AUTH rules now consume the new `StateIntermX` facts instead. After the successful exchange of the IKE_INTERMEDIATE a new keymat is derived and used to protect the following exchanges. Another addition with the

IKE_INTERMEDIATE messages is the modification of the authentication values that now includes hashes of the two new messages. Different than with the IKE_SA_INIT, both agents sign the sent and the received message of the new exchange. The rest of the model remains identical to the original IKEv2 model.

6.3.2 Adapted Properties

The properties and lemmata are modeled as closely to the regular IKEv2 model as possible in order to not introduce mistakes in the model. For the augmented protocol the complexity of the proofs increased with the additional state required to model the IKE_INTERMEDIATE exchange. Nevertheless all proofs can be solved without manual interaction using only Tamarin's automatic solver.

For these new properties the attacker is allowed to circumvent the original Diffie-Hellman key exchange at any given time, with the help of an oracle. Instead the effects of a compromise of the second key exchange are observed, which models the post-quantum key exchange. The new attacker model is equivalent to the old one but with an attacker leveraging a quantum computer potent enough to apply Shor's algorithm and break the classical key exchange.

6.3.2.1 Validity of the Model

In order to proof the final state, which is equivalent to the protocol performing a successful authenticated key exchange with intermediate exchange, can actually be reached in the model, the two `exists_session` lemmata are adapted to the new model.

`exists_session` verifies that a single coherent session can be negotiated between the two peers using a two key exchanges, one in the IKE_SA_INIT and one in the IKE_INTERMEDIATE exchange. The lemma itself is The Tamarin definition is given in Listing 6.14.

```
lemma exists_session: exists-trace
    "Ex A B spi ck #i #j .
        IKeys(A, B, spi, ck) @ i
        & RConfirm(A, B, spi, ck) @ j
        & i < j"
```

Listing 6.14: Session exists, PQ-IKEv2

The lemma itself is identical to the one used for the non-hybrid IKEv2 proof. It still applies because the new model requires the IKE_INTERMEDIATE step to be taken in order to get to the IKE_AUTH and consequently satisfy the `IKeys` and `RConfirm` properties.

The same is true for the second validity proof called `exists_two_sessions`. The lemma shows that it is possible to complete more than one full run of the protocol in a single instance of the model. The property is relevant because many network protocol attacks require leaking some data in one session and applying the gained knowledge in a new session, e.G. replay attacks. For Tamarin, to find these kinds of attacks, the attacker model must be able to interact with more than one protocol run. The Tamarin definition is given in Listing 6.15.

```
lemma exists_two_sessions: exists-trace
    "Ex A B spi spi2 ck ck2 #i #j #i2 #j2 .
        IKeys(A, B, spi, ck) @ i
        & RConfirm(A, B, spi, ck) @ j
        & i < j
        & IKeys(A, B, spi2, ck2) @ i2
        & RConfirm(A, B, spi2, ck2) @ j2
        & i2 < j2
        & not (ck=ck2)"
```

Listing 6.15: Two Sessions exists, PQ-IKEv2

The PQ-IKEv2 model satisfies both the `exists_session` and `exists_two_sessions` lemmata.

6.3.2.2 Authentication

Aliveness

For the three lemmas that verify the different levels of authentication minor changes were necessary to adapt them to the new model. The first property called *aliveness* shows that a complete run of the protocol for A with B is only possible if B has been running the protocol before. For this property the only modification required was to explicitly state that the values of `keymat` may differ for the two peers.

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

Listing 6.16: Aliveness, PQ-IKEv2

Weak Agreement

For the *weak agreement* property the attacker is now allowed to use the `RevDH` oracles that were explicitly handled in the original lemma. Instead the new action fact `RevDHQ()` is checked, which corresponds to the attacker leaking one of the new post-quantum keys exchanged with IKE_INTERMEDIATE.

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

Listing 6.17: Weak Agreement, PQ-IKEv2

Agreement For the final authentication property, simply called *agreement*, the modification looks similar. For this property both of the weaker notions of authentication must be true. Further, the agreed upon set of data, in this case `keymat` must be the same for both parties and A must act as the initiator and B as the responder.

```
lemma agreement[use_induction]:
    "All spi A B keymat #i .
        Completed(spi, A, 'initiator', B, keymat) @ #i
        & not (Ex #k . RevLtk(B) @ #k)
        & not (Ex #k . RevDHQ(A) @ #k)
        & not (Ex #k . RevDHQ(B) @ #k)
    ==> (Ex #j .
        Completed(spi, B, 'responder', A, keymat) @ #j)"
```

Listing 6.18: Agreement, PQ-IKEv2

The results of the analysis for the different *authentication* properties are identical, which is why they are summarized here. Note that the order of the notions is strictly hierarchical, which means *agreement* implies *weak agreement* as well as *aliveness*.

No compromised keys When no keys are compromised the IKEv2 protocol achieves *agreement*.

Compromised post-quantum ephemeral keys When only the post-quantum ephemeral keys are compromised, the protocol achieves *agreement*

Compromised static keys When only the initiator's static key is compromised, the protocol achieves *agreement*. When the responder's static key is compromised, the attacker can "impersonate" the role of the responder and the protocol fails to achieve any of the notions of *authentication*.

Compromised post-quantum ephemeral & static keys When the post-quantum ephemeral keys and the initiator's static key are compromised, the protocol provides *agreement*. When the responder's static key is compromised, no form of *authentication* is no longer given.

The results are identical to those of the non-post-quantum IKEv2 protocol with the only difference that the new additional key exchange has taken the place of the classical key exchange with the old attacker model. It can be concluded that the additional post-quantum key exchange succeeds to provide *authentication* even against a quantum computer attack, when only attacks on the key exchange are considered.

6.3.2.3 Consistency

Consistency

Again, consistency is defined as follows: "When two agents A and B each complete a run of the protocol, and I acts as the initiator and thinks it talks to R and R acts as responder and thinks it talks to I, then they both agree on a identical set of session variables."

The Tamarin definition, found in Listing 6.19 remains unchanged from the one used in the IKEv2 analysis.

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

Listing 6.19: Consistency, PQ-IKEv2

No compromised keys When no keys are compromised the PQ-IKEv2 protocol always achieves *consistency*.

Compromised post-quantum ephemeral keys When only the ephemeral keys of a single party or of both parties are compromised, PQ-IKEv2 achieves *consistency*.

Compromised static keys When only the initiator's static key is compromised, the protocol achieves *consistency*. When the responder static key has been compromised, an attacker can "impersonate" the other party resulting in a loss of *consistency*.

Compromised post-quantum ephemeral & static keys When both ephemeral keys and the initiator static key are compromised, the protocol achieves *consistency*. When the responder's static key is compromised, the attacker can break *consistency*

The analysis of IKEv2 showed a lack of consistency in the case of a compromised responder authentication key. The model shows that the same attack is possible with the post quantum hybrid IKEv2 protocol. This result is expected as the post-quantum modification did not change the protocols authentication mechanism.

6.3.2.4 Key Secrecy

Key Secrecy

Only initiator and responder know the keys of a successful run of the protocol, or inverted: After a successful run of the protocol the attacker must not know the secret session key material. The Tamarin lemma's definition is the same as for the IKEv2 protocol.

```
lemma key_secrecy[reuse]:
    "All spi A B role keymat #i .
        Completed(spi, A, role, B, keymat) @ #i
        & not (Ex #j . RevLtk(B) @ #j)
        & not (Ex #j . RevDHQ(A) @ #j)
        & not (Ex #j . RevDHQ(B) @ #j)
    ==> not (Ex #j . K(keymat) @ #j)"
```

Listing 6.20: Key Secrecy, PQ-IKEv2

No compromised keys When no keys are compromised, the PQ-IKEv2 protocol achieves *key secrecy*.

Compromised post-quantum ephemeral keys The compromise of a single one of the ephemeral keys leads to a loss of *key secrecy*, the attacker can generate the shared key material from the public nonces and the shared Diffie-Hellman secret.

Compromised static keys When a participants key is leaked, the same participant still achieves *key secrecy*. When the key of a trusted peer is compromised, the attacker can impersonate said peer and break *key secrecy*/

Compromised post-quantum ephemeral & static keys When any ephemeral key is leaked, the protocol does not provide *key secrecy*.

6.3.2.5 Identity Hiding

The protocol parts concerning *identity hiding* have not changed in any meaningful way with the integration of the IKE_INTERMEDIATE exchange. The expected result is that the PQ-IKEv2 protocol achieves *identity hiding* against a quantum computer attacker, unless any of the post-quantum ephemeral or the initiator's long term static key have been compromised.

```
lemma identity_hiding:
    "All spi A B id #i .
        Completed(spi, B, 'responder', A) @ #i
        & IdentityLearnt(id) @ #i
        & not (Ex #k . RevLtk(A) @ #k)
        & not (Ex #k . RevDHQ(A) @ #k)
        & not (Ex #k . RevDHQ(B) @ #k)
    ==> not (Ex #j .  K(id) @ #j)"
```

Listing 6.21: Identity Hiding, PQ-IKEv2

No compromised keys With no compromised keys the PQ-IKEv2 protocol achieves *identity hiding* for the responder.

Compromised post-quantum ephemeral keys A compromised ephemeral key breaks *identity protection*.

Compromised static keys With the responder's static key compromised the protocol achieves *identity hiding*. With compromised initiator static key the attacker can learn the responder's identity and break *identity hiding*.

Compromised post-quantum ephemeral & static keys The compromise of a single ephemeral key leads to a loss of *identity protection*, the same goes for a compromise of any ephemeral and any static key at the same time.

The analysis confirms the assumption that the hybrid PQ-IKEv2 exchange has no negative impact on the *identity protection* property of IKEv2 but adds security against a quantum attacker.

6.4 Summary

The formal analysis of the security of the IKEv2 and the modified hybrid PQ-IKEv2 protocol allow two conclusions:

First, the security properties of the original IKEv2 handshake have been confirmed and have passed the validity test against the results of former formal verification aided analysis of the same protocol, such as [CHH$^+$17] and [MD$^+$97]. For each of the security properties, the Tamarin verification comes to the same results as previous publications, ultimately confirming once more that the IKEv2 protocol satisfies all desired security properties unless one or more of the secret keys have been compromised. The analysis even independently found a minor violation of the authentication property in the case of a compromised ephemeral key which has also been known before. These identical results at the same time confirm the validity of the presented IKEv2 model. The second, novel, result of the analysis is that, using the same model as the analysis of the original IKEv2 protocol with only minor additions, the proposed quantum-resistant PQ-IKEv2 protocol achieves *exactly* the same security properties that IKEv2 has, but against the stronger quantum computer attacker model.

The proposed protocol fulfills all of the security requirements found in Section 4.2.2, which leaves only the usability requirement open for evaluation.

7 Proof of Concept

After the previous Chapter 6 has shown that the designed protocol succeeds to hold the security properties of the original protocol and adds additional protection against an attacker leveraging a quantum computer potent enough to calculate Shor's algorithm for cryptographically used numbers, some of the desired properties are best evaluated in practical experiments.

This Chapter introduces a proof of concept implementation of the quantum-resistant IKEv2 protocol presented in Chapter 5. The PQ-IKEv2 implementation is then experimentally evaluated to show that quantum-resistant key exchange schemes are not only secure, but also practical in common VPN use cases.

7.1 Implementation

The PQ-IKEv2 implementation is based on the OpenBSD operating system's IKEv2 daemon, simply named **iked** [Flo19a]. It provides an implementation of the basic IKEv2 protocol, support for extensions is sparse but exists for some of the more useful extensions (IKE fragmentation is supported). The daemons main advantage is its compact code base. As a result, the daemon's main loop is well-structured and has a low complexity which makes it easily extendable and a perfect starting point for a proof of concept implementation of the post-quantum key exchange.

7.1.1 OpenBSD iked

The iked daemon is running completely in user land. In order to still be able to initiate ESP SAs which have to be enforced by the kernel, it uses the IETF standardized PFKEY2 interface which is described in [MMP98]. The daemon at first start saves the IPsec flows in the kernel. IKE SAs are all negotiated and managed in user space, as they don't require any kernel interaction. Child SAs resulting from the IKEv2 negotiation again are transmitted to the kernel via PFKEY. If the kernel finds incoming or outgoing traffic matches any of the IPsec flows but no SA is available, it sends out a push notification called *AQUIRE*. The notification instructs user-land daemons to negotiate a Child SA for the corresponding flow. This way each implementations responsibilities and their interworkings are well defined.

When the daemon is first started it will, depending on its configuration, determine what SAs have to be initiated and then send the initial IKE_SA_INIT messages as specified. An example configuration is given in Listing 7.1.

The configuration reads as follows: Use IKEv2 to construct an ESP tunnel between the nets *10.0.0.0/24* and *10.0.1.0/24*. The IKEv2 peer can be reached at *10.0.1.0*. For the IKE SA,

```
ikev2 esp active from 10.0.0.0/24 to 10.0.1.0/24 \
        peer 10.0.1.0 \
        ikesa group x25519
```

Listing 7.1: A simple iked configuration

use a Diffe-Hellman group value of *x25519* (which corresponds to the ECDH group of the same name). In this case the keyword *active* instructs the IKE daemon to actively start the security association by sending out an IKE_SA_INIT request. While this configuration only defines one IKE SA, it is possible to have multiple IKE SAs with different peers that are managed by a single iked instance. A more detailed documentation of the configuration language with multiple examples for common IKEv2 setups is documented in [Flo19b].

Once the initial messages have been sent, the daemon enters the main loop that processes incoming IKEv2 messages asynchronously using the *event* library [Pro19]. Incoming messages are handled by the *ikev2_recv()* function. A visualization of the functions inner workings in the form of a systems diagram is given in Figure 7.1. An incoming message first gets its header parsed and checked for correctness. The SPI found in the header is used to look up a corresponding SA from the Security Association Database (SAD). The resulting SA is used to check the further header information for correctness. This includes a comparison of the messages *EXCHANGE_ID* with the SA's internal state, to make sure no SA can get authenticated twice, as well as the messages sequence number which should be higher than the one in the SA, otherwise the message is a replay. If the header passes all checks, the rest of the message containing only the payloads goes on to the daemons payload parsing facility. The payloads are parsed into a temporary structure one after another. If the encrypted *SK* payload is encountered, it is decrypted and integrity checked using the keys from the SA and if everything is correct the contents go back into the payload parser. It should be noted that up to this point, there has been no change to the SA or any other non-temporary internal state. This is crucial to the security of the implementation as otherwise an attacker could send broken or erroneously encrypted messages and use them to change internal state like the SAs sequence number, which would hinder the daemon from receiving future valid messages from the SA's peer because their sequence number is lower than what the daemon expects ext. After the message was parsed successfully and all checks have passed, the temporary contents structure together with the SA go into the exchange handler, which applies all changes resulting from the exchange to the SA. This may include an update of the keys, e.G. after the *CREATE_CHILD_SA* exchange or simply a change of the internal state. After the changes are applied the updated SA is stored back into the SAD and if needed a response message is sent out.

The implementation of the post-quantum key exchange also requires the IKE_INTERMEDIATE extension to be implemented. As the exchange cannot easily be implemented without a use case, the following will treat them as a single addition to the protocol. In other words: the implemented IKE_INTERMEDIATE can in this form solely be used for the hybrid key exchange. The minor changes for the IKE_INTERMEDIATE are the addition of a new notify payload that is sent in the IKE_SA_INIT to negotiate support for the extension, as well as the addition of a new exchange ID. The actual exchange implementation is added in the new `ikev2_ike_intermediate()` function. The function implements the *Handle*

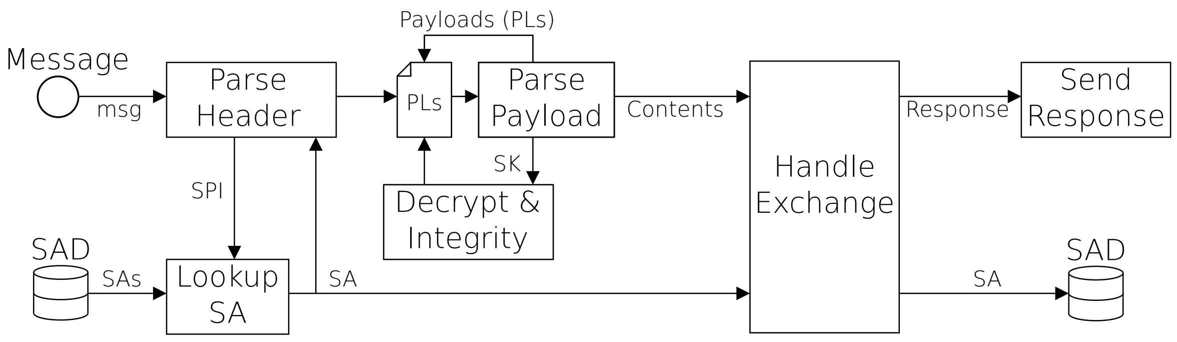

Figure 7.1: Design of iked's message receive function

Exchange step as shown in Figure 7.1 and expects the message contents to contain a single key and a new nonce (as defined in the PQ-IKEv2 protocol), for which the payload parser can be reused. The handler then stores the new values in the SA, eventually sends out the response in case it acts as the responder, and then updates the session keys for the used SA. As the hybrid key update takes the same form as the IKE SA rekeying, the implementation can reuse the `ikev2_sa_keys()` function with minor modifications. Another change introduced in the IKE_INTERMEDIATE exchange is the modification of the authentication signature that is sent in IKE_AUTH. For each IKE_INTERMEDIATE exchange, both messages are added to the signed octets of both peers, so different than with the IKE_SA_INIT the received and sent messages are signed. To enable the post-quantum key exchange, some new configuration options for the seven additional key exchanges were introduced labeled *ake1* to *ake7*. Listing 7.2 shows a similar configuration as seen before, only this time the IKE SA uses a hybrid key exchange with the x25519 ECDH curve in the IKE_SA_INIT and the brainpool512 curve in an additional IKE_INTERMEDIATE exchange.

```
ikev2 esp active from 10.0.0.0/24 to 10.0.1.0/24 \
        peer 10.0.1.0 \
        ikesa group x25519 ake1 brainpool512
```

Listing 7.2: A simple hybrid iked configuration

7.1.2 Open Quantum Safe

Now that the implementation supports the hybrid key exchange as specified in Chapter 5, the only thing missing for a real life evaluation of the post-quantum hybrid key exchange is the addition of actual quantum-resistant key exchange methods. Chapter 3 introduced two libraries that provide some of the NIST post-quantum key exchange candidates with a common interface and build system. The proof-of-concept implementation uses the *liboqs* [SM16] from the Open Quantum Safe project because it has a more convenient build system that could easily be ported to the OpenBSD operating system which the implementation runs on. The library allowed to extend *iked* with the following selection of new PQKE options:

- BIKE1_L1
- BIKE1_L3
- BIKE1_L5
- BIKE2_L1
- BIKE2_L3
- BIKE2_L5
- BIKE3_L1
- BIKE3_L3
- BIKE3_L5
- Kyber512
- Kyber1024
- NewHope-512-CCA
- NewHope-1024-CCA
- FrodoKEM-640-AES
- FrodoKEM-640-SHAKE
- FrodoKEM-976-AES
- FrodoKEM-976-SHAKE
- FrodoKEM-1344-AES
- FrodoKEM-1344-SHAKE
- Sidh-p503
- Sidh-p702
- Sike-p503
- Sike-p702

The downside of *liboqs* over *libpqcrypto* is the limitation to only some of the 17 proposed schemes, but the library offers several different configurations for each scheme. Moreover the implemented schemes include KEs selected across all categories described in Chapter 2.2.3, which allows a detailed comparison of the usability of each approach.

The schemes are implemented in the daemon as new transforms for the Diffie-Hellman group type, which allows to use them interchangeably with the classical Diffie-Hellman and ECDH exchanges. An example configuration with a hybrid post-quantum key exchange using the "NewHope" scheme is presented in Listing 7.3

```
ikev2 esp active from 10.0.0.0/24 to 10.0.1.0/24 \
        peer 10.0.1.0 \
        ikesa group x25519 ake1 newhope512
```

Listing 7.3: A hybrid post-quantum iked configuration

A more detailed list of the new options comes in the *manual page* of `iked.conf(5)` included with the implementation.

7.2 Benchmarks

To evaluate whether the presented protocol is actually practical for use in real world settings, the implementation was tested with different IKEv2 configurations. The test environment is straightforward, two hosts "A" with IP *10.0.1.33* and "B" with *10.0.1.34* are connected directly via Ethernet. Both hosts run an up-to-date snapshot of OpenBSD from October 2019, including the patched *iked* daemon and a self-built *liboqs* library. The host configuration for "A" can be seen in Listing 7.4.

```
set fragmentation
ikev2 "test" esp active from 10.0.1.33 to 10.0.1.34 \
    peer 10.0.1.34 \
    ikesa group x25519 [ake1 ...]\
    psk "sharedkey"
```

Listing 7.4: The Test Setup Configuration

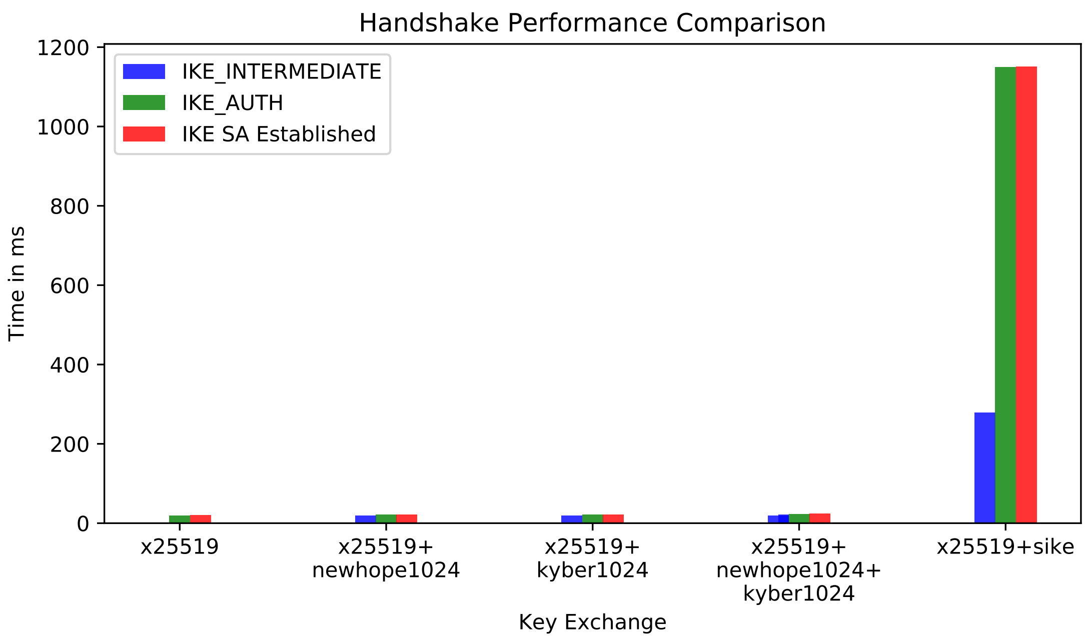

Figure 7.2: Key Exchanges in Comparison: SIKE

The config of host “B” looks identical with the only exception that the IPs are switched and “B” is passive instead of active, meaning only “A” will initiate the handshake. Fragmentation is enabled and as the setup uses IPv4 addresses IKEv2 packets will be fragmented to a maximum size of 548 Bytes. For simplicity, all tests use pre-shared key authentication, which should not make a difference between post-quantum and non-post-quantum IKEv2.

The first measurement compares a classical *x25519* ECDH exchange with four different hybrid exchanges added on top of *x25519*. These include *newhope1024* and *kyber1024*, a combination of both, and the isogeny based *sikep751*. Each of the exchanges was performed 100 times, and then averaged. The result can be seen in Figure 7.2. Measurements were taken on the initiator side at four different points in the protocol: Before the first IKE_SA_INIT message is sent, to know when the exchange started, before the IKE_AUTH request is sent and after receiving the IKE_AUTH reply when the IKE_SA is activated. Each bar represents the time from the start to the measurement, so the blue bar shows the time until the IKE_INTERMEDIATE was sent, the green bar the time until the IKE_AUTH was sent and the red bar the total time for the exchange.

What becomes obvious at first sight is that the hybrid exchange using *sike* seems to perform much worse than the other candidates in the benchmark. In total it takes over 1 second until the IKE SA is activated, while the curve25519 alone, *kyber* and *newhope* take less than 25 ms. This result is even more interesting when considering fragmentation: *sike* has the smallest public keys of the compared schemes at 644 Byte resulting in the IKE_INTERMEDIATE messages being sent in two fragments each. Meanwhile the *x25519+newhope1024+kyber1024* does two IKE_INTERMEDIATE exchanges with five and four fragments respectively. Fragmentation and key size seems to only have a limited negative impact in this test setup. On the other hand as seen in Section 4.1.1 *sike's* computational performance is known to be

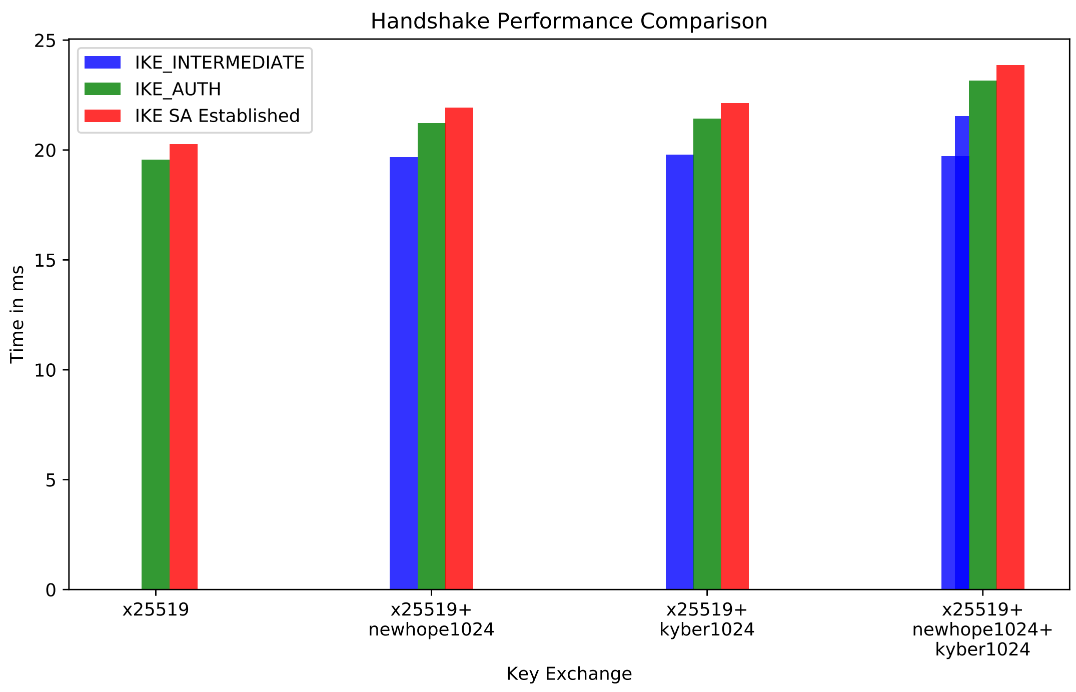

Figure 7.3: Key Exchanges in Comparison: Kyber and NewHope

by far the worst of all NIST candidates which the test confirms. With key exchange times of over one second the scheme is not a realistic Diffie-Hellman replacement in any latency critical use-case.

Figure 7.3 gives a closer look at the four better performing configurations which all stay under 25 ms for the full key exchange. It seems that most time for each exchange is spent for the setup of the initial message and the SA initialization, seen in the height of the left bar of each key exchange. The actual key exchange round-trips seem to make only a small difference in comparison. As expected the non-hybrid exchange doing only *x25519* is the fastest at roughly 20 ms closely followed by *x25519+newhope1024* and *x25519+kyber1024* which both finish at approximately 22 ms with *newhope* being slightly faster and last the exchange combining all of the before mentioned, *x25519+newhope1024+kyber1024* at slightly less than 24 ms. It is also the only configuration with two overlaying IKE_INTERMEDIATE bars, and the difference from blue to blue and from blue to green seems to be almost identical, meaning the two post-quantum exchanges take approximately the same time at roughly 2 ms each, which is also the difference between non-post-quantum exchange and each of the two-exchange hybrid combinations.

The results show that the presented protocol can indeed provide a realistic countermeasure against quantum-attackers adding merely 10% latency to the initial handshake depending on the chosen cryptographic key exchange method. Even a three-way hybrid handshake, which will rarely be necessary except for highly confidential data, costs only 20

8 Conclusion

The time is ticking against classical exchange methods, as a breakthrough in quantum computing might happen any day. Even worse, the IKEv2 key exchange is vulnerable to an ex-post attack where the attacker saves the encrypted communication today and breaks the key exchange handshake in the future, once a powerful enough quantum computer exists. A compromise of the IKEv2 key exchange leads to a leak of the symmetric ESP encryption keys which allow the attacker decrypt the following IPsec communication.

While there presently exists no quantum computer able to factor numbers in sizes as they are used in asymmetric cryptography, the research for novel quantum-resistant key exchange methods has produced a number of cryptographic schemes existing today that are designed to be secure against both classical and quantum computers. Some major problems of those schemes is the uncertainty when it comes to their actual security, because most lack proper cryptographic review, as well as their unique constraints when it comes to performance or key size.

This work aimed to find a way to integrate post-quantum cryptographic key exchange algorithms into the IKEv2 key exchange protocol. Therefore it has thoroughly has analyzed and categorized most of the post-quantum key exchange schemes to better understand their specific constraints, and then compared them to the abilities and constraints of the IKEv2 key exchange protocol to find what is necessary to integrate the new algorithms in a protocol originally designed to be used only with a single Diffie-Hellman key exchange. What became clear is that none of the post-quantum schemes is established enough to even be considered secure against classical computers. Instead of using a post-quantum scheme in place of the original Diffie-Hellman exchange, the solution lies in the use of a hybrid key exchange of a well understood classical key exchange such as ECDH with the x25519 curve combined with one or even more post-quantum schemes. The idea is that in case the post-quantum scheme turns out to be broken in the future the protocol falls back to the security of a strong classical exchange, whil, if the post-quantum scheme holds up, an attacker can break ECDH but not the additional PQKE, combining the best of both worlds. In addition it has become clear that the selection of a single post-quantum scheme will not be enough, instead the protocol should be agnostic of what key exchange scheme it transports. The resulting requirement is called *crypto agility*, the ability to freely exchange a cryptographic algorithm with another.

A further analysis of the security of the IKEv2 protocol lead to a clear definition of four security properties that the protocol must fulfill: *Authentication*, *Consistency*, *Key Secrecy* and *Identity Hiding*. These properties are used to have a basis for comparing the IKEv2 protocol as is with a modified protocol to support PQKEs.

With the requirements being clear, the biggest change required to the protocol was clearly the integration of a hybrid key exchange. For all protocol design choices several options were laid out, partly compiled from related work, partly original ideas first introduced in this work.

After a detailed discussion of the advantages and disadvantages of each approach, a single solution was chosen resulting in the definition of a new quantum-resistant IKEv2 protocol named PQ-IKEv2. With the requirements guiding the protocols design choices, the result is a protocol implementing a hybrid key exchange which allows to freely combine any of the defined key exchange mechanisms of the IKEv2 protocol, clearly separating the two problem domains of hybrid KE and post-quantum KE. In the proposed protocol, post-quantum KEs behave exactly like their non-post-quantum counterparts, and both can be freely combined and exchanged. The hybrid classical and post-quantum exchange is only one of the allowed combinations.

The design of the protocol has resulted in the PQ-IKEv2 which is meant to implement a secure hybrid and possibly post-quantum key exchange, but whether the required protocol changes have actually preserved the claimed security of the IKEv2 protocol is unclear. The problem with security is that it can not easily be measured and thus is hard to compare. In order to get a better understanding of the security of the new protocol, this work resorts to a method that has been used in the design of another recent IETF protocol: TLS 1.3, which has been formally analyzed with a symbolic network protocol proofer called Tamarin in each major iteration of the proposed RFC draft. This process has helped to eliminate attack vectors early in the design stages of the protocol and turned out to be a useful addition to the informal expert review process an IETF protocol has to go through.

In order to compare the original IKEv2 and the PQ-IKEv2 protocols for their security properties, this work has modeled both protocols in the Tamarin proofer and translated the properties found in the security analysis lemmas the automatic proofer can validate or negate. Previous formal analysis of the IKEv2 protocol allowed a validation of the findings against and thus a baseline for the correctness of the implemented model. As the PQ-IKEv2 protocol handshake is a superset of the original, the IKEv2 model could easily be extended while the almost identical lemmata could be reused. As a result it was shown that the IKEv2 protocol actually provides the security it claims, which is in line with previous work on the subject, and more importantly that the added hybrid key exchange provides the same security, even against an attacker that can break the original key exchange. The result is groundbreaking as it is the first time formal analysis has been used in the design of an IKEv2 protocol extension.

After the design of the protocol was final, with the security properties proven to be fulfilled a last uncertainty was whether the protocol was actually usable in a real world setting, which for a key exchange mostly means does it perform fast enough or does the added hybrid key exchange add a considerably to the latency of the exchange. The protocol was therefore implemented in the open IKEv2 implementation used in the OpenBSD operating system called *iked*. The post-quantum cryptography schemes are provided by the *liboqs* software library which provides various quantum-resistant scheme implementations. A benchmark of several different schemes has shown that the latency impact from the hybrid key exchange protocol turns out to be considerably smaller than the performance difference between the post-quantum schemes themselves, making a combination of three different well-chosen schemes more than 40 times faster than a combination of x25519 and the isogeny based *SIKE* scheme. Even more, the hybrid key exchange using the promising structured-lattice based *NewHope* and *kyber* schemes add only 2 milliseconds to the overall key exchange, which is only a 10% increase in latency, which shows that the proposed construction is actually suited to be used in real world settings without major caveats.

This work has shown that it is actually possible and even practical to make the IKEv2 protocol quantum-resistant without losing security or usability of the protocol. The required tools and implementations exist today and the only thing stopping us from widely deploying quantum-resistant communication protocols are the missing quantum-resistant cryptography standards, which are actively being worked on. It is unclear when NIST's standardization process will come to an end, but the theoretical work for IKEv2 integration is mostly done and once the NIST "winners" are being announced they can easily be integrated into the protocols.

Another possible future problem with the availability of quantum computers is not solved by this protocol, which is quantum-resistant authentication. The PQ-IKEv2 protocol still uses classical authentication, because it is considered much harder to break than the key exchange. In order to break authentication, it must be broken in real time and there is no ex-post attack as for the key exchange. It is uncertain whether and when quantum computers will be potent enough and available for an attacker to perform a real time authentication attack, but once a quantum computer exists that can run Shor's algorithm this problem should also be considered.

The results of this work in the form of source code of the Tamarin IKEv2 and PQ-IKEv2 model as well as the protocol implementation are freely available in the hopes that they will be used as basis for a formal analysis in the design process of future IKEv2 extensions. A future work may also extend the model with the existing non-implemented protocol states such as the optional EAP-only authentication or the cookie challenge, which would help to further confirm the security of the protocol as a whole, not only the reduced initial handshake this work has considered.

Nomenclature

AES	Advanced Encryption Standard
AH	Authentication Header
BSD	Berkley Software Distribution
CVP	Closest Vector Problem
D-H	Diffie-Hellman Group (in IKEv2)
DH	Diffie-Hellman
DOS	Denial of Service
DSA	Digital Signature Algorithm
EAP	Extensible Authentication Protocol
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ENCR	Encryption Algorithm
ESN	Extended Sequence Numbers
ESP	Encapsulating Security Payload
FTP	File Transfer Protocol
HDR	Header
HMAC	Hash-Based Message Authentication Code
HTTP	Hypertext Transfer Protocol
ICV	Initialization Vector
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
INTEG	Integrity Alorithm
IP	Internet Protocol
IPsec	Security Architecture for the Internet Protocol
KA	Key Agreement
KE	Key Exchange

KEM	Key Encapsulation Mechanism
KEX	Key Exchange
LWE	Learning With Errors
MAC	Message Authentication Code
MTU	Maximum Transmission Unit
NIST	National Institute for Standards and Technology
PAD	Peer Authorization Database
PFS	Perfect Forward Secrecy
PK	Public Key
PKCS	Public Key Crypto System
PLD	Payload
PPT	Probabilistic Polynomial Attacker
PQC	Post-Quantum Cryptography
PQKE	Post-Quantum Key Exchange
PRF	Pseudorandom Function
PSK	Pre-Shared Key
RAM	Random Access Memory
RFC	Request For Comments
RLWE	Ring Learning With Errors
RSA	Rivest-Shamir-Adleman Cryptosystem
SA	Security Association
SAD	Security Association Database
SHA	Secure Hash Algorithm
SI	Supersingular Isogeny
SIDH	Supersingular Isogeny Diffie-Hellman
SIGMA	Sign and Mac Protocol
SIKE	Singular Isogeny Key Encapsulation
SPD	Secure Policy Database
SPI	Secure Parameter Index
SSL	Secure Socket Layer
SVP	Shortest Vector Problem

TLS	Transport Layer Security
TS	Traffic Selector
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network

List of Figures

Bibliography

[ABB$^+$15] Augot, Daniel ; Batina, Lejla ; Bernstein, Daniel J. ; Bos, Joppe ; Buchmann, Johannes ; Castryck, Wouter ; Dunkelman, Orr ; Güneysu, Tim ; Gueron, Shay ; Hülsing, Andreas u. a.: Initial recommendations of long-term secure post-quantum systems. In: *Available at pqcrypto. eu. org/docs/initial-recommendations. pdf* (2015)

[ABD$^+$15] Adrian, David ; Bhargavan, Karthikeyan ; Durumeric, Zakir ; Gaudry, Pierrick ; Green, Matthew ; Halderman, J A. ; Heninger, Nadia ; Springall, Drew ; Thomé, Emmanuel ; Valenta, Luke u. a.: Imperfect forward secrecy: How Diffie-Hellman fails in practice. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* ACM, 2015, S. 5–17

[BCNS15] Bos, Joppe W. ; Costello, Craig ; Naehrig, Michael ; Stebila, Douglas: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: *2015 IEEE Symposium on Security and Privacy* IEEE, 2015, S. 553–570

[BDH$^+$18] Basin, David ; Dreier, Jannik ; Hirschi, Lucca ; Radomirovic, Saša ; Sasse, Ralf ; Stettler, Vincent: A formal analysis of 5G authentication. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* ACM, 2018, S. 1383–1396

[CC19] Campagna, Matt ; Crockett, Eric: Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS) / Internet Engineering Task Force. Version: Mai 2019. https://datatracker.ietf.org/doc/html/draft-campagna-tls-bike-sike-hybrid-01. Internet Engineering Task Force, Mai 2019 (draft-campagna-tls-bike-sike-hybrid-01). – Internet-Draft. – Work in Progress

[CHH$^+$17] Cremers, Cas ; Horvat, Marko ; Hoyland, Jonathan ; Scott, Sam ; Merwe, Thyla van d.: A comprehensive symbolic analysis of TLS 1.3. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* ACM, 2017, S. 1773–1788

[CK01] Canetti, Ran ; Krawczyk, Hugo: Analysis of key-exchange protocols and their use for building secure channels. In: *International Conference on the Theory and Applications of Cryptographic Techniques* Springer, 2001, S. 453–474

[CK02] Canetti, Ran ; Krawczyk, Hugo: Security analysis of IKE's signature-based key-exchange protocol. In: *Annual International Cryptology Conference* Springer, 2002, S. 143–161

[CLM$^+$18] Castryck, Wouter ; Lange, Tanja ; Martindale, Chloe ; Panny, Lorenz ; Renes, Joost: CSIDH: an efficient post-quantum commutative group action.

In: *International Conference on the Theory and Application of Cryptology and Information Security* Springer, 2018, S. 395–427

[Cre11] Cremers, Cas: Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In: *European Symposium on Research in Computer Security* Springer, 2011, S. 315–334

[Den03] Dent, Alexander W.: A designer's guide to KEMs. In: *IMA International Conference on Cryptography and Coding* Springer, 2003, S. 133–151

[DH76] Diffie, Whitfield ; Hellman, Martin: New directions in cryptography. In: *IEEE transactions on Information Theory* 22 (1976), Nr. 6, S. 644–654

[DM17] Donenfeld, Jason A. ; Milner, Kevin: Formal verification of the wireguard protocol / Technical Report. 2017. – Forschungsbericht

[DY83] Dolev, Danny ; Yao, Andrew: On the security of public key protocols. In: *IEEE Transactions on information theory* 29 (1983), Nr. 2, S. 198–208

[EST10] Eronen, Pasi ; Sheffer, Yaron ; Tschofenig, Hannes: *An Extension for EAP-Only Authentication in IKEv2*. RFC 5998. http://dx.doi.org/10.17487/RFC5998. Version: September 2010 (Request for Comments)

[Flo19a] Floeter, Reyk: *OpenBSD iked(8) manual page*. https://man.openbsd.org/iked.8. Version: 6.5, Juli 2019

[Flo19b] Floeter, Reyk: *OpenBSD iked.conf(5) manual page*. https://man.openbsd.org/iked.conf.5. Version: 6.5, Juli 2019

[FS07] Fu, David E. ; Solinas, Jerome: *ECP Groups For IKE and IKEv2*. RFC 4753. http://dx.doi.org/10.17487/RFC4753. Version: Januar 2007 (Request for Comments)

[Goo16] Google: *Experimenting with Post-Quantum Cryptography*. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html. Version: 2016

[Goo19] Google: *Transparency Report HTTPS encryption on the web*. https://transparencyreport.google.com/https/overview?hl=en. Version: 2019

[Gro96] Grover, Lov K.: A fast quantum mechanical algorithm for database search. In: *arXiv preprint quant-ph/9605043* (1996)

[Har02] Harkins, Dan: Design Rationale for IKEv2 / Internet Engineering Task Force. Version: Februar 2002. https://datatracker.ietf.org/doc/html/draft-ietf-ipsec-ikev2-rationale-00. Internet Engineering Task Force, Februar 2002 (draft-ietf-ipsec-ikev2-rationale-00). – Internet-Draft. – Work in Progress

[HPS98] Hoffstein, Jeffrey ; Pipher, Jill ; Silverman, Joseph H.: NTRU: A ring-based public key cryptosystem. In: *International Algorithmic Number Theory Symposium* Springer, 1998, S. 267–288

[KCB97] Krawczyk, Hugo ; Canetti, Ran ; Bellare, Mihir: HMAC: Keyed-hashing for message authentication. (1997)

[KE06] Korhonen, Jouni ; Eronen, Pasi: *Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol*. RFC 4739. http://dx.doi.org/10.17487/RFC4739. Version: November 2006 (Request for Comments)

[Ken05a] Kent, Stephen: *IP Authentication Header*. RFC 4302. http://dx.doi.org/10.17487/RFC4302. Version: Dezember 2005 (Request for Comments)

[Ken05b] Kent, Stephen: *IP Encapsulating Security Payload (ESP)*. RFC 4303. http://dx.doi.org/10.17487/RFC4303. Version: Dezember 2005 (Request for Comments)

[KHN+14] Kaufman, Charlie ; Hoffman, Paul E. ; Nir, Yoav ; Eronen, Pasi ; Kivinen, Tero: *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. http://dx.doi.org/10.17487/RFC7296. Version: Oktober 2014 (Request for Comments)

[Kiv11] Kivinen, Tero: *Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)*. RFC 6467. http://dx.doi.org/10.17487/RFC6467. Version: Dezember 2011 (Request for Comments)

[Kiv16] Kivinen, Tero: Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation. (2016)

[KK16] Kremer, Steve ; Künnemann, Robert: Automated analysis of security protocols with global state. In: *Journal of Computer Security* 24 (2016), Nr. 5, S. 583–616

[Kra03] Krawczyk, Hugo: SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: *Annual International Cryptology Conference* Springer, 2003, S. 400–425

[KS15] Kivinen, Tero ; Snyder, Joel: *Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)*. RFC 7427. http://dx.doi.org/10.17487/RFC7427. Version: Januar 2015 (Request for Comments)

[Lan16] Langley, Adam: *Imperial Violet CECPQ1 results*. https://www.imperialviolet.org/2016/11/28/cecpq1.html. Version: 2016

[Lan18] Langley, Adam: *Imperial Violet CECPQ2 results*. https://www.imperialviolet.org/2018/12/12/cecpq2.html. Version: 2018

[Low97] Lowe, Gavin: A hierarchy of authentication specifications. In: *Proceedings 10th Computer Security Foundations Workshop* IEEE, 1997, S. 31–43

[McE78] McEliece, Robert J.: A public-key cryptosystem based on algebraic. In: *Coding Thv* 4244 (1978), S. 114–116

[MD+97] Moedersheim, S ; Drielsma, PH u. a.: *AVISPA Project Deliverable D6. 2: Specification of the Problems in the High-Level Specification Language (2003)*. 1997

[Mea99] Meadows, Catherine: 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, 1999, S. 216–231

[MI88] Matsumoto, Tsutomu ; Imai, Hideki: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: *Workshop on* *the Theory and Application of of Cryptographic Techniques* Springer, 1988, S. 419–453

[MLLL$^+$12] Martín-López, Enrique ; Laing, Anthony ; Lawson, Thomas ; Alvarez, Roberto ; Zhou, Xiao-Qi ; O'brien, Jeremy L.: Experimental realization of Shor's quantum factoring algorithm using qubit recycling. In: *Nature Photonics* 6 (2012), Nr. 11, S. 773

[MMP98] Metz, Craig ; McDonald, Daniel L. ; Phan, Bao: *PF_KEY Key Management API, Version 2.* RFC 2367. http://dx.doi.org/10.17487/RFC2367. Version: Juli 1998 (Request for Comments)

[Nie86] Niederreiter, Harald: Knapsack-type cryptosystems and algebraic coding theory. In: *Prob. Control and Inf. Theory* 15 (1986), Nr. 2, S. 159–166

[NIS16] NIST: *Pqc-Forum Key Establishment for PQC algorithms.* https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/oc3Xl0ZKLGI/discussion. Version: 2016

[NIS19] NIST: *Post-Quantum Cryptography Round 2 Submissions.* https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions. Version: 2019

[NJ16] Nir, Yoav ; Josefsson, Simon: *Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement.* RFC 8031. http://dx.doi.org/10.17487/RFC8031. Version: Dezember 2016 (Request for Comments)

[PQC19] PQCRYPTO: *libpqcrypto cryptographic software library.* https://libpqcrypto.org/. Version: Juli 2019

[Pro19] Provos, Niels: *OpenBSD event(3) manual page.* https://man.openbsd.org/event.3. Version: 6.5, Juli 2019

[SFG19] Steblia, Douglas ; Fluhrer, Scott ; Gueron, Shay: Design issues for hybrid key exchange in TLS 1.3 / Internet Engineering Task Force. Version: Juli 2019. https://datatracker.ietf.org/doc/html/draft-stebila-tls-hybrid-design-01. Internet Engineering Task Force, Juli 2019 (draft-stebila-tls-hybrid-design-01). – Internet-Draft. – Work in Progress

[Sho99] Shor, Peter W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In: *SIAM review* 41 (1999), Nr. 2, S. 303–332

[Sin01] Singer, Ari: NTRU Cipher Suites for TLS / Internet Engineering Task Force. Version: Juli 2001. https://datatracker.ietf.org/doc/html/draft-ietf-tls-ntru-00. Internet Engineering Task Force, Juli 2001 (draft-ietf-tls-ntru-00). – Internet-Draft. – Work in Progress

[SK05] Seo, Karen ; Kent, Stephen: *Security Architecture for the Internet Protocol.* RFC 4301. http://dx.doi.org/10.17487/RFC4301. Version: Dezember 2005 (Request for Comments)

[SM16] Stebila, Douglas ; Mosca, Michele: Post-quantum key exchange for the internet and the open quantum safe project. In: *International Conference on Selected Areas in Cryptography* Springer, 2016, S. 14–37

[SMCB12] Schmidt, Benedikt ; Meier, Simon ; Cremers, Cas ; Basin, David: Automated analysis of Diffie-Hellman protocols and advanced security properties. In: *2012 IEEE 25th Computer Security Foundations Symposium* IEEE, 2012, S. 78–94

[Smy14] Smyslov, Valery: *Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation.* RFC 7383. http://dx.doi.org/10.17487/RFC7383. Version: November 2014 (Request for Comments)

[Smy19] Smyslov, Valery: Intermediate Exchange in the IKEv2 Protocol / Internet Engineering Task Force. Version: Juni 2019. https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-intermediate-01. Internet Engineering Task Force, Juni 2019 (draft-ietf-ipsecme-ikev2-intermediate-01). – Internet-Draft. – Work in Progress

[SW15] Smyslov, Valery ; Wouters, Paul: *The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2).* RFC 7619. http://dx.doi.org/10.17487/RFC7619. Version: August 2015 (Request for Comments)

[TTg+19] Tjhai, C. ; Tomlinson, M. ; grbartle@cisco.com ; Fluhrer, Scott ; Geest, Daniel V. ; Garcia-Morchon, Oscar ; Smyslov, Valery: Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2) / Internet Engineering Task Force. Version: Januar 2019. https://datatracker.ietf.org/doc/html/draft-tjhai-ipsecme-hybrid-qske-ikev2-03. Internet Engineering Task Force, Januar 2019 (draft-tjhai-ipsecme-hybrid-qske-ikev2-03). – Internet-Draft. – Work in Progress

[Val17] Valence, Henry de: *SIDH in Go for quantum-resistant TLS 1.3.* https://blog.cloudflare.com/sidh-go/. Version: 2017

[Zim15] Zimmer, Dipl-Inf E.: Post-Quantum Kryptographie für IPsec. (2015)