State of the art

In this section we provide the necessary concepts about the following topics:

- The IKEv2 protocol
- Post-quantum cryptography
- Existing means to equip IKE with quantum security
- Existing techniques to implement a (somewhat) lighter version of IKE

1.1 The Internet Key Exchange (IKE) protocol

The main purpose of the IKEv2 protocol, as described in [AD-3], is to set up and maintain "IKE Security Associations" (SA in short) between two endpoints. Informally, creating an SA means agreeing on a set of cryptographic algorithms, as well as a shared secret (to be used as the secret key), in a secure and efficient manner. Here, by secure we mean that the established connection shall provide provide authentication, data integrity and confidentiality.

The endpoint starting the communication is referred to as *initiator* while the other endpoint is, consequently, called *responder*. The protocol messages are grouped into *exchanges* and each exchange is constituted by pairs of request/reply messages.

In its base version, IKEv2 provides four types of exchanges:

1. `IKE_SA_INIT`: establish the connection, negotiate the cryptographic suite, exchange DH material;
2. `IKE_AUTH`: provide authentication;
3. `CREATE_CHILD_SA`: create child SAs or rekey existing SAs;
4. `INFORMATIONAL`: notify about exceptional events (e.g., closing an SA).

The first exchange is not encrypted, while all the subsequent ones are, using the algorithms that have been negotiated in phase 1. An example of IKEv2 communication involving all the four types of exchanges and one retransmission is reported in Figure 2. Notice that the `IKE_SA_INIT` and `IKE_SA_AUTH` are mandatory, must be completed in the correct order and before any other exchange.

Apart from these four exchanges, IKEv2 can use a fifth exchange, called `IKE_INTERMEDIATE`. This type of exchange, introduced in [**RFC9242**], has been initially introduced to let the shared key be derived from multiple key exchanges (i.e., not only from the material exchanged in the `IKE_SA_INIT` exchange). It turns out that, as described in [RD-10], the use of this exchange is fundamental to achieve post-quantum cryptography without using pre-shared keys. More details on such a feature are provided in the remainder of this section.

1.2 Post-Quantum cryptography

Post-quantum cryptography studies cryptographic schemes with quantum security, i.e., with the ability to withstand attacks from adversaries equipped with quantum computers. This way, one takes into account attackers that are capable of running quantum algorithms (such as Shor or Grover) and, hence, are capable of breaking some of the fundamental primitives we use nowadays. For this reason, the current state-of-the-art cryptographic algorithms must be updated/replaced with newer algorithms.

As it is well known, the algorithms for symmetric primitives (such as private key encryption schemes and hash functions) do not need to be modified. Indeed, even though algorithms such as Grover provide an exponential speed-up with respect to classical algorithms, best quantum attacks remain exponential time. For instance, looking a symmetric key encryption scheme with keys of length n, a quantum CPA attack would run in time $O(2^{n/2})$ while a classical CPA attack would take $O(2^{n})$ operations. Hence, conservatively, doubling the key size would be enough to restore the security level. Notice that this a very conservative consideration as it completely neglects the cost of running, in practice, quantum algorithms. Many authors argue that, in practice, quantum attacks on symmetric algorithms will never be faster than classical ones so that, for symmetric primitives, nothing should be modified (not even increasing the key size!).

For what concerns asymmetric primitives, the situation is completely the opposite. Indeed, these algorithms are based on the hardness of solving some number theoretic problems such as the integer factorization or the discrete logarithm (in a finite and commutative group). While all known classical algorithms can solve these problems in exponential time, quantum algorithms such as Shor's run in polynomial time.[1] In practice, this means that quantum attackers can break such schemes very rapidly and, to make such algorithms slow, one would require huge public keys and ciphertexts. For this reason, the major challenge post-quantum cryptography is facing consists in finding efficient and secure alternatives for asymmetric cryptographic primitives. This can be achieved by considering mathematical problems which are different from those employed in classical algorithms and which, more importantly, cannot be solved efficiently by quantum algorithms.

[1] For instance, Shor's algorithm solves the integer factorization problem in time $O(n^3)$, where n denotes the binary size of the input number.

Despite being a rather new discipline, post-quantum cryptography has rapidly emerged as a fundamental need and many industries and governments has already started the migration process to post-quantum cryptography. The most famous and important example of this effort is represented by the NIST PQC competition, started back in 2017 and currently in the fourth and final round. Four algorithms, called *winners*, have already been selected and included in the standards while three algorithms, referred to as *alternate*, are still under evaluation [AD-1, AD-2].

NIST standardized algorithms Some macroscopic characteristics of the four winners in the NIST PQC competition are listed in Table 9. The first three algorithms have already been included in the standards FIPS 203, 204 and 205 with official names, respectively, ML-KEM, ML-DSA and SLH-DSA. The standard for the signature scheme based on Falcon has yet to appear.

Type	**Scheme**	**Secret key size (bytes)**	**Public key size (bytes)**	**Ciphertext/ Signature size (bytes)**	**Security Category**
KEM	Kyber-512	1632	800	768	I
	Kyber-768	2400	1184	1088	III
	Kyber-1024	3168	1568	1568	V

	Dilithium3-SHAKE	-	1952	3293	III
	Dilithium5-SHAKE	-	2592	4595	V
	Dilithium2-AES	-	1312	2420	I
	Dilithium3-AES	-	1952	3293	III
	Dilithium5-AES	-	2592	4595	V
	Dilithium5-AES	-	2592	4595	V
SIGNATURE	SPHINCS+128s	64	32	7856	I
	SPHINCS+128f	64	32	17088	I
	SPHINCS+192s	96	48	16224	III
	SPHINCS+192f	96	48	35664	III
	SPHINCS+256s	128	64	29792	V
	SPHINCS+256f	128	64	49856	V
	FALCON- 512	1888	897	666	I
	FALCON-1024	2840	1793	1280	V

Table 9: Parameters of the standardized NIST PQC algorithms.

Classical vs Post-Quantum As the table highlights, these algorithms come with some significant drawbacks since the size of public keys, ciphertexts and signatures gets generically larger with respect to classical algorithms. Moreover, as we will later show, running post-quantum algorithms is computationally more expensive than executing classical algorithms. Hence, as a general statement, we can say that transitioning to PQC comes with a non negligile cost for what concerns practical applications.
Indeed, PQC currently cannot provide (in a secure and practical way) all the functionalities which classical cryptography provides. An example, which is very relevant for the use cases we consider in this project, is that of the Diffie-Hellman key exchange, since we do not know about a post-quantum counterpart. This may have important consequences in protocols such as IKEv2, as they strongly depend on the Diffie-Hellman key exchange to establish session keys.

The most common post-quantum replacement for the Diffie-Hellman key exchange is represented by KEMs which, however, work in a different way. Indeed, KEMs are used to let a user derive a key and communicate it to the other end; some authors call this algorithm *key transport*, to emphasize that the key is chosen by a single user and then transferred to the other end. See Figure 3 for a representation of how a KEM operates. In the figure, we have indicated by (sk, pk) the key-pair associated with the KEM, and by k the key which is encapsulated by the responder and transported to the initiator.[2]

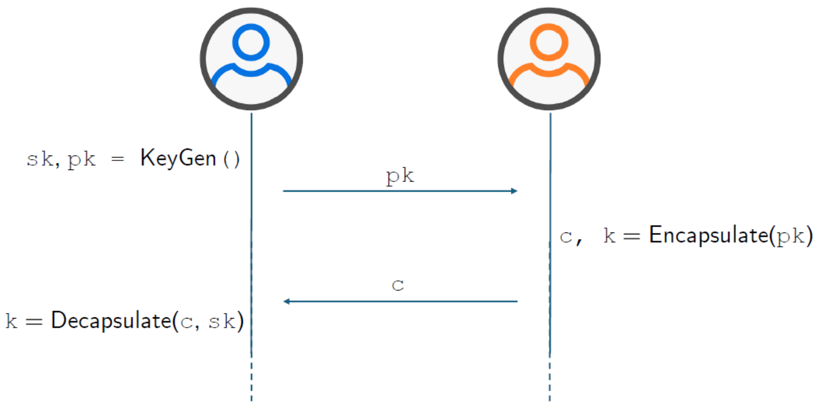

Figure 3: Algorithmic flow of a KEM

1.3 Post-Quantum IKE

There exist different strategies to make IKEv2 quantum secure, with the two most popular options being:

- **pre-shared keys** [RD-7]: together with the keying material derived from the DH key-exchange, the SA secret key depends also on some secret string which must be shared between the endpoints before the `IKE_SA_INIT` takes place. Obviously, the entropy of such strings (i.e., the number of bits with which they are represented) should be large enough.
- **multiple key-exchanges** [RD-10]: in the `IKE_SA_INIT` phase, multiple key exchanges or KEMs take place. All the exchanged material is then employed to derive the shared session key. Obviously, the additional exchanges must be post-quantum. These additional exchanges are performed through the `IKE_INTERMEDIATE` exchange [**RFC9242**].

Both solutions are called extensions since they do not compromise interoperability with devices that do not support them. Notice that, with pre-shared keys, quantum security comes from the fact that quantum attackers can only try Grover's algorithm which, however, takes exponential time. Hence, security of the resulting protocol is not based on the presumed hardness of some mathematical problem [RD-8]. Moreover, this choice would not lead to a significant computational overhead, with respect to the classical IKEv2. However, the feasibility of this solution is questionable because of some practicability issues, e.g.,

[2] The key k is derived, using a PRF or a hash function, from random quantities which are sampled inside the algorithm and later returned as one of the outputs.

How are pre-shared strings distributed? How are pre-shared keys updated?

Relying on multiple key exchanges allows to exploit at most what PQC offers, as the above questions are naturally solves using post-quantum KEMs to transfer keys. However, this comes with some drawbacks, as more bandwidth and computations are required. Obviously, in case public key certificates are employed for authentication, obviously, post-quantum signatures shall be used.

IKEv2 with multiple exchanges When multiple key exchanges are employed, the SA secret key is derived as `s = Hash(k, k′)`, where `k` is the key resulting from the IKE_SA_INIT (i.e., the one obtained with a classical DH exchange) and `k′` is the one encapsulated through a post-quantum KEM. [3] See Figure 4 for a representation of the situation. Notice that post-quantum security comes from the fact that the two exchanges are executed independently, hence, an attacker breaking the first exchange (for instance, retrieving `a` from `A`) does not obtain any information about `sk` or `k′`.

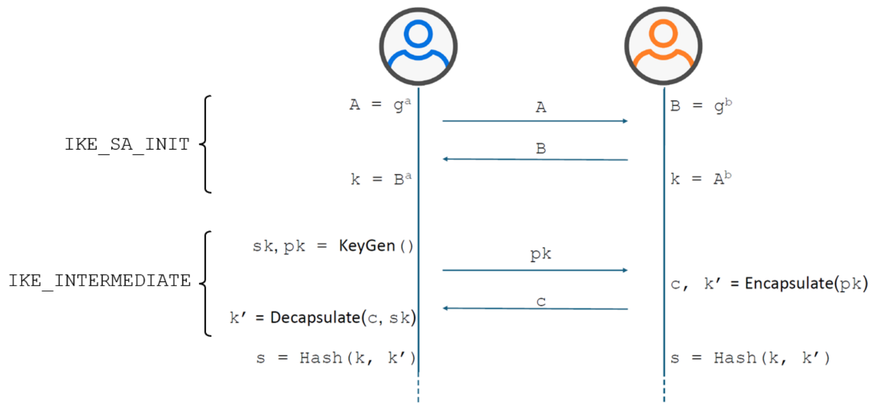

Figure 4: Representation of key derivation in IKEv2, using the `IKE_INTERMEDIATE` exchange with KEMs

The situation depicted in Figure 4 can be generalized, e.g.,

- the classical DH exchange can be avoided: the `IKE_SA_INIT` messages are sent without the DH values;

[3] The derivation of `s` is a bit more complicated as, for instance, many keys are derived, one employed for encryption, one for authentication, etc. Still, conceptually, there is no meaningful difference with our description, which is much simpler.

- multiple KEMs can be employed in `IKE_INTERMEDIATE`; in the end, the derived secret `s` would be computed using a PRF on all the encapsulated keys.

1.4 Minimal IKE

This version of IKEv2 [RD-15], which we will call Minimal IKE, refers to a somewhat lighter version of IKEv2 which is achieved, essentially, by sacrificing some of the functionalities that a fully fledged implementation of IKEv2 would guarantee. Indeed, [RD-15] describes how to implement an IKEv2 initiator which, among others, lacks the following features: NAT traversal, IKE SA rekey, Child SA rekey, multiple Child SAs, deleting Child / IKE SAs, Configuration payloads, Extensible Authentication Protocol (EAP) authentication, COOKIEs, etc.

In practice, the initiator in Minimal IKE has only the following functionalities:

- ability to initiate the `IKE_INITIATE` and `IKE_AUTH` exchanges;
- minimal replies to incoming requests, e.g., a `CREATE_CHILD_SA` response containing the `NO_ADDITIONAL_SAS` error notify, and an empty `INFORMATIONAL` response for an `INFORMATIONAL` request;
- send a `delete_IKE_SA` notification.

In practice, this allows to greatly simplify the implementation of the IKEv2 protocol. For instance, the initiator does not need to keep track of its own message IDs, nor those of incoming requests. Also, the initiator does not need to keep track of created SAs since, by design, the possibility of creating child SAs is deprecated: whenever it wants to connect to some other endpoint, it must create a new SA and delete it when the no more messages need to be exchanged.

Notice that Minimal IKE comes with some requirements which may be challenging to fulfil, especially when one considers post-quantum version of the IKE protocol. For instance, [RD-15] suggests to avoid public key certificates as they are rather expensive in terms of computational cost. Moreover, [RD-15] suggests to rely on pre-shared keys as this is the solution coming with the lowest overhead. As a trade-off solution, [RD-15] suggests to consider raw public keys.

1.5 Header compression technique

?????

2 Simulations and considerations

To quantify how impactful is turning IKEv2 into a post-quantum protocol, we have simulated the creation of an SA using the IKEv2 implementation provided by Strongswan (https://www.strongswan.org/). [**Qualcosa sul fatto che Strongswan è una libreria affidabile, mantenuta, certificata, ecc.**]

Strongswan already provides some post-quantum functionalities which, currently, are contained in the forthcoming version 6.0, which is already available but only as a test version (https://github.com/strongX509/docker/tree/master/pq-strongswan#readme).

For what concerns post-quantum cryptography, Strongswan relies on the Open Quantum Safe library (https://github.com/open-quantum-safe/liboqs). The supported algorithms are all the standardized ones, plus Falcon (yet to appear in an official standard), the alternate ones (apart from ClassicMcEliece, whose public keys are too large) and some exceptional algorithms (such as FrodoKEM).

In the remainder of this section, we consider how switching to post-quantum cryptography affects the performances of IKEv2 implemented with Strongswan.

2.1 Setup

For all our simulations, we have considered the following setup:

- Docker containers to implement the communicating endpoints (that is, the initiator and responder)
- **Caratteristiche del dispositivo: RAM, PROCESSORE, etc.**
- **Sistema operativo**
- Altro di importante?

To derive a proper comparison between the best of both worlds (i.e., classical vs post-quantum), we have considered three configurations, whose security is equivalent to NIST category 1 (that is, as hard as breaking AES-128):

- Configuration A:
    - Only classical security
    - AES-128 in CTR mode
    - SHA-256
    - ECDH with group: [**Nome gruppo**]
    - ECDSA signatures

- Configuration B:
    - Post-quantum security
    - AES-256 in CTR mode
    - SHA-512
    - Kyber_L1
    - Dilithium_L2

- Configuration C:

  - Post-quantum security
  - AES-256 in CTR mode
  - SHA-512
  - Kyber_L1
  - Falcon_L2

We expect configuration C to provide the smaller messages, as Falcon signatures and public keys are smaller than those of Dilithium. Nonetheless, configuration B is probably more convenient from the hardware point of view, as Kyber and Dilithium are essentially based on the same mathematical objects and most of the subroutines employed by one algorithm can be used by the other. [**TO BE VERIFIED**]

2.2 Numerical results

[**DATA TO BE INCLUDED**]

2.3 Considerations

[**SOME COMMENTS**]