SCTE•ISBE CABLE-TEC EXPO® VIRTUAL EXPERIENCE » OCTOBER 12-15 2020

2020 Fall Technical Forum SCTE·ISBE • NCTA • CABLELABS®

DOCSIS® PKI: A Proposal for a Next-Generation Quantum-Resistant Infrastructure

Using Composite Crypto for Post-Quantum PKI Transitioning

A Technical Paper prepared for SCTE•ISBE by

Massimiliano Pala
Principal Security Architect
CableLabs
858 Coal Creek Cir., Louisville, CO 80027
+1 (303) 661-3334
m.pala@cablelabs.com

Table of Contents

List of Figures

Title **Page Number**

1. Introduction

The broadband industry has been relying on public-key cryptography (PKC) to provide secure and strong authentication across its networks and devices. In particular, the DOCSIS standard [Doc31, Doc40] uses X.509 [Itu509] certificates to verify that a device is a legitimate entity that is authorized to join the network—for example, a cable modem or a Remote PHY (R-PHY) node [Rphy1]. The choice of using digital certificates and public-key infrastructures (PKIs) to protect DOCSIS identities has resulted in a scalable and easy-to-deploy key management system for the entire industry.

Although the DOCSIS PKI has been a success story over the past 20 years (it is one of the largest PKIs ever deployed worldwide), things are changing rapidly on both the security side and the broadband industry side.

On the security side, new advancements in traditional and non-traditional computing are threatening our ability to use traditional public-key and key-exchange (KEX) algorithms. On the network infrastructure side, new zero-trust architectures are being designed that require software and hardware entities to securely authenticate to each other (and encrypt traffic) in a distributed environment.

This paper describes our proposal for a backward-compatible quantum-resistant trust infrastructure (or PKI) for the broadband industry. Specifically, our work focuses on the practical aspects of deploying a quantum-resistant trust infrastructure by leveraging our idea—namely, the composite cryptography mechanism [Com20].

The paper is organized as follows: Section 2 provides a description of the quantum threat for the various parts of a PKI; Section 3 describes the composite crypto solution and its two building blocks (i.e., `CompositeKey` and `CompositeSignature`); Section 4 describes how to practically deploy composite crypto in PKIs; Section 5 provides considerations surrounding the use of secure elements and hardware security modules (HSMs); and Section 6 describes a deployment proposal for securing the DOCSIS PKI. Finally, Section 7 provides our conclusions and envisioned future work.

2. Trust Infrastructures and the Quantum Threat

Deploying real security is difficult, and security engineers rely on basic cryptographic primitives to make sure protocols operate as intended and data is accessed only by authorized entities. Deploying real security is even more difficult when uncertainty around the efficacy of your basic tools is at risk. Unfortunately, we are living in a period of great cryptographic "uncertainty," where the advancements in both traditional and quantum computing pose serious threats that may impact the possibility to provide secure access and data privacy not only for the broadband industry, but across the Internet.

Recently, advancements in quantum computing suggest that the possibility to break PKC might be closer than initially thought. Specifically, the work of Craig Gidney and Martin Ekerå [GE19] improves the efficiency of quantum computers to perform code-breaking calculations, thus reducing the required resources by orders of magnitude. For example, in 2015, researchers estimated that a billion quantum bits or qubits (aka q-bits) would be required to factor 2048-bit keys due to the need to use noise-reduction codes that require significant extra qubits themselves. With the recent advancement from Gidney and Ekerå, the number of required qubits is reduced to only 20 million.

Another example of the fast-paced rhythm of innovation occurring in the quantum space is the fact that researchers have already moved away from studying qubits to building quantum logic gates. Many

experts theorize that at the current speed of innovation, we will have a quantum computer within 20 years—a very short period of time in the cryptography space.

The takeaway message is alarming for all of us: Governments, military and security organizations, banks, medical facilities and everybody else will have no options to secure their data against powerful quantum computers.

The situation is even more alarming when considering that there are no complete or general solutions today that allow for securing all aspects of a PKI—not just signatures or certificates. In particular, regardless of the specific standard or technology used in a PKI (e.g., X.509, PGP), no deployments currently cover the possibility of using multiple algorithms in a combined fashion to further secure the infrastructure in case one or more algorithms are deemed compromised. Our approach addresses this limitation by leveraging a recursive construct for both public-key signatures and keys that can be applied to every aspect of the PKI lifecycle management.

On the quantum-safe cryptography standardization front, things are moving forward as NIST recently announced the beginning of Round 3 in its selection of a quantum-resistant cryptography standard that began few years ago [Nist20].

Because the new standard will specify one or more quantum-resistant algorithms for digital signatures, public-key encryption and key generation, the selection of finalists comprises various classes of algorithms and mathematical properties. In particular, of the original 69 proposals submitted for Round 1, only 26 made it to Round 2. For Round 3, only 8 candidates were selected. Four of the candidate crypto-systems provide public-key encryption, while the remaining four are digital signatures schemes.

The selected key encapsulation mechanisms (KEMs) that provide public-key encryption are:

- **Classic McEliece.** This KEM is the result of a merger between the Classic McEliece and NTS-KEM. This work is a code-based KEM based on the original cryptosystem from 1978. The cryptosystem has never gained much interest in the cryptographic community; however, its resistance to Shor's algorithm makes it a good candidate for post-quantum standardization. Classic McEliece, which uses the binary Goppa code, comes with very large public keys but the smallest ciphertext of any other solution. Although this performance profile might not be the best fit for the general protocols we use over the Internet today, its stable specifications combined with a long history of cryptoanalysis make it an appealing choice for some applications.

- **CRYSTALS-Kyber.** This algorithm is based on the presumed hardness of the module learning with errors (MLWE) problem. The scheme has excellent all-around performance for most applications. It also enables relatively straightforward adjustment of the performance-versus-security tradeoff by varying module rank and noise parameters. NIST regards this scheme as one of the most promising KEMs for standardization.

- **NTRU**. This is a structured lattice-based KEM that has an established adoption history because variations of this KEM have already been standardized by IEEE [Ntru09] and ANSI [Ntru10] organizations. Although NTRU has a small performance gap in comparison with Kyber and SABER, its longer history was an important factor in NIST's decision because of less risk of unexpected intellectual property claims. An important characteristic of NTRU is the fact that it relies on a different problem than MLWE; thus, it provides diversity in the set of structured lattice-based KEMs for the finalists.

- **SABER.** This KEM is based on a variation of MLWE—namely, the Module Learning With Rounding (MLWR), in which rounding from one modulus to a smaller second one replaces the addition of small errors. In general, SABER provides good performance for general-purpose applications. One of the areas that NIST encourages more research on is side-channel analysis for the non-Number Theoretic Transform (non-NTT) style of multiplication that is unique to SABER. Together with NTRU and Kyber, SABER is one of the most promising KEMs selected for Round 3.

For digital signatures, the following are considered as finalists:

- **CRYSTALS-Dilithium.** This is a lattice-based signature scheme that relies on the MLWE hardness and the module short integer solution (MSIS). One of the advantages of CRYSTALS-Dilithium over its main competitor, Falcon, is the simpler implementation due to the use of the same modulus and ring for all parameter sets and samples. Overall, Dilithium has a good balance in terms of key and signature sizes and performs well for key generation, signing and verification, making it one of the strongest candidates for standardization as it performs well in real-world experiments.

- **Falcon.** The Falcon signature scheme is lattice-based and uses the "hash and sign" paradigm. Although this scheme is more complex to implement than Dilithium, it offers the smallest public key and signature sizes. From a key generation point of view, Falcon is slower than other candidates, but the overall strong performance makes it a good fit for existing Internet protocols and applications.

- **Rainbow.** This scheme is very different from the previous two. Rainbow is a multivariate signature scheme with an unbalanced oil and vinegar (UOV) construct. Rainbow provides fast signing and verification, along with very short signatures. The downside of this scheme is the extremely large public key sizes. Some issues with the security claims and the fact that NIST prefers algorithms with royalty free licensing put this algorithm at the bottom of the list.

- **GeMSS.** This is the second multivariate signature scheme that uses the "big field" paradigm. Although it offers the smallest signatures of any other schemes and provides a fast verification algorithm, the large size of public keys is the limiting factor for adoption, especially in low-end devices where signing can be very slow. Also, the large size of public keys makes it impractical when used with TLS or SSH without protocol changes. This algorithm is still in consideration, in case developments in Round 3 show the Rainbow scheme to not be suitable for standardization.

NIST also selected eight alternative algorithms whose evaluation will continue after the first selection for the standard. These eight alternative algorithms either might need more time to mature or are tailored to more specific applications. For example, the Sphinx+ scheme is considered very secure but also very conservative in its design. Because this translates into higher bandwidth and slower performance than the selected primary candidates, Sphinx+ is being considered only as a backup option for standardization. The review process for Round 4 will continue after Round 3 ends, and eventually some of these alternative algorithms could become part of the standard at a later date. The status of the NIST selection process is available in the NIST Internal Report 8309 [NISTIR].

2.1. PKI Building Blocks

To convey which parts of a trust infrastructure will be affected by the use of quantum computing, we provide a classification of algorithms or "building blocks" that are used to protect both authentication and user data and show how they might be impacted by the quantum threat.

Specifically, when it comes to the various types of building blocks, we shall differentiate between public-key algorithms for authentication (e.g., RSA, ECDSA), KEX algorithms (e.g., Diffie-Hellman, Elliptic-Curves Diffie-Hellman), hashing algorithms (e.g., SHA-256, SHA-3) and encryption algorithms (e.g., AES).

Public-Key Algorithms. Public-key algorithms are primarily used to authenticate (and sometimes encrypt) data. Algorithms like RSA or ECDSA are the most common when used in conjunction with X.509 digital certificates. The main difference between RSA and ECDSA, besides the underlying mathematical properties, is in the size of the cryptographic overhead (or bandwidth) and key-generation complexity. In particular, when compared through the performance lens, ECDSA has a clear advantage, especially when deployed in small or computationally limited devices. When compared through the security lens, though, further considerations are due. One interesting feature of RSA is its ability to use different key lengths without changing the algorithm. This allows, for example, RSA cryptosystems to increase the size of the public/private keypair to adjust to increased security risks due to advancements in computing and crypto research. ECDSA, instead, does not support this feature: Once the curve is selected (e.g., NIST's Secp256r1 curve), the key size cannot be changed, and to increase the security of the system, new curves (e.g., NIST's Secp521r1 curve) must be supported (even for validation only).

Key-Exchange Algorithms. This class of algorithms has been recently revamped because of the effort, in the TLS space, to deploy perfect forward secrecy (PFS). Specifically, the use of finite-field Diffie-Hellman (DH) and Elliptic-Curve Diffie-Hellman (ECDH) has been required by the latest TLS specifications [RFC 8446] to overcome the security limitations of the RSA-based KEX mechanism and decouple authentications from key exchanges. KEX algorithms are at the core of protocols like TLS to securely derive the encryption keys used after the negotiation phase.

Hashing Algorithms. This class of algorithms is at the core of the authentication process in modern PKIs. In fact, to authenticate any type of data (e.g., a certificate or simply a document), the data must be "summarized" to make the signing process efficient and more generic (i.e., providing the same operational approach when working with different algorithms). Because of this, breaking the properties of the hashing algorithms can be quite devastating for a public-key cryptosystem.

Encryption Algorithms. When it comes to securing data from unauthorized access, encryption algorithms provide the necessary building blocks via the use of securely exchanged (via the authentication process) encryption keys. The ability to successfully attack these algorithms can bypass any authentication process used during the transfer of encryption keys and therefore grant an attacker direct access to the data.

In the rest of this section, we provide a summary of the threats for each class of algorithms and how (and if) the quantum threat is significant in that space.

2.2. Modern Cryptography and the Quantum Threat

2.2.1. Public-Key Cryptography

When it comes to public-key algorithms such as RSA or ECDSA, the threat of quantum computers being able to "guess" (factor large numbers or solve the discrete logarithm problem) private keys is comparable for both cryptosystems. In fact, PKC uses mathematical algorithms to generate complex keys, thus making the code to reverse-engineer the private component from the public one statistically very hard. Different public-key systems can use different algorithms, as long as they are based on mathematical problems that are easy to put into place but difficult to reverse-engineer. For instance, any computer can multiply two extremely large prime numbers, but factoring the result is nearly impossible—at least, it would be for a classical machine. Although only few classes of algorithms (so far) have been identified to be more performant on quantum computers, factoring large numbers is one of them. Specifically, Shor's quantum factorization algorithm could be easily used to break this type of cryptosystem.

The optimization that is possible on a quantum computer is due to the fact that it should be able to use the properties of quantum mechanics (e.g., entanglement) to probe for patterns within a huge number without having to examine every digit in that number. Because cracking both RSA and EC ciphers actually involves finding patterns in huge numbers, quantum computers can perform the inverse operation at practically the same speed as the forward one. For example, while on a conventional computer, finding a pattern for an EC cipher would take $2^{N/2}$ steps—where N is the number of bits in the key. On a quantum computer, the number of steps would be in the order of only N/2!

RSA, ECC and DH are examples of cryptosystems that will not be secure against an adversary with access to a quantum computer.

2.2.2. Hashing Algorithms

Hashing algorithms give us a chance to breathe a little easier. As explained in the previous section, some problems with a special structure (e.g., factorization for RSA, discrete log for EC) typically fall in the category of problems for which quantum computers can reduce the task to finding the period of some function, which is surprisingly not difficult to solve on a quantum computer.

However, for unstructured problems, quantum computers seem to provide "only" some non-trivial quadratic speedup, via the use of Grover's algorithm [Gro96]. Simply put, this indicates that a hash function with 256 bits of security today would still have 128 bits of security in a post–quantum computing era.

2.2.3. Encryption Algorithms

Symmetric encryption, and more specifically AES-256, is believed to be quantum-resistant. Quantum computers are not expected to be able to reduce the attack time enough to be effective if the key sizes are large enough. Also in this case, the speedup from Grover's algorithm allows quantum computers to perform exhaustive searches in the square root of the classical time, rendering classical attacks more expensive than generic ones in most cases. To achieve 128 bits of security in a post–quantum computing scenario (equivalent to AES-128 today), primitives providing 128 bits of security (e.g., AES) need to have a key length of at least 256 bits. In other words, the speedup in the quantum exhausting search provided by Grover's algorithm is the main reason why the current recommendations for encryption with post–quantum computing security mandate for AES [Dr99] with a 256-bit key.

If you are using AES in your systems in 2020, you should favor AES-256 over AES-128. This is true for software environments, but it is especially important when chipsets and hardware deployment is involved; to make sure these devices can take you over the "quantum hump," support for AES-256 is required [Aes16, Aes20].

Ultimately, as with the hashing algorithms case, modern symmetric encryption algorithms are not affected as badly by quantum computing as public-key ones as long as we deploy larger keys (e.g., AES-256).

2.3. Algorithm Agility to the Rescue

Algorithm agility is at the core of best practices when it comes to cryptosystems today. Algorithm agility refers to the possibility of substituting cryptographic algorithms (and associated data structures) as needed, without having to change protocol messages or main data structures.

Fortunately, the X.509 standard, used across the Internet and in the broadband industry, provides the possibility to extend the generic data structure to integrate new algorithm-dependent ones in order to identify public keys and signatures. Technically, this is achieved by coupling the crypto data structures with an algorithm identifier that can correctly validate and process the associated data structure(s) across different types of objects used in PKIs (e.g., X.509 certificates, Online Certificate Status Protocol [OCSP] responses, Certificate Revocation Lists [CRLs]).

For example, look at how public keys are encoded in X.509 certificates:

```
AlgorithmIdentifier  ::=  SEQUENCE  {
        algorithm            OBJECT IDENTIFIER,
        parameters           ANY DEFINED BY algorithm OPTIONAL  }

SubjectPublicKeyInfo  ::=  SEQUENCE  {
        algorithm            AlgorithmIdentifier,
        subjectPublicKey     BIT STRING  }
```

Here, `SubjectPublicKeyInfo` comprises an `AlgorithmIdentifier` that identifies the cryptographic algorithm and associated parameters, as well as a `subjectPublicKey`, which is a `BIT STRING` [RFC 5280]. The value of `subjectPublicKey` is the Distinguishing Encoding Rule (DER) encoding of the public-key structure as defined for the specific algorithm used.

For example, Section 2.3.1 of [RFC 3279] defines the contents of `SubjectPublicKeyInfo` and how to encode the `RSAPublicKey` structure whose DER representation is to be used for the value of `subjectPublicKey`.

Our solution leverages the algorithm agility feature and defines a new algorithm identifier that encodes, recursively, a set of one or more `subjectPublicKey` data structures to encode multiple keys with different algorithms, as discussed in the next section.

3. A Composite Crypto-Based Solution

The cable industry, as well as the larger Internet community, is faced with the very difficult task of addressing the quantum threat early in order to continue to securely authenticate network devices and users by switching to different algorithms when needed. On top of that, the broadband industry faces the additional challenge of handling this future transition while relying on fielded devices (e.g., CMs, R-PHY/R-MACPHY nodes) that might be expensive to replace or update.

Although the selection for standard post-quantum algorithms has not been finalized yet, we have been actively looking at how to deploy such algorithms for the broadband industry when they become available. In particular, we looked at how to protect the integrity of the most vulnerable parts of our DOCSIS trust infrastructure first (from a quantum-threat perspective)—that is, the root and intermediate CA. We focused on these assets first because they are valuable targets that would allow for an attacker to generate new valid entities for the ecosystem. Once the integrity of the upper levels of the hierarchy are secured, devices can be updated to support the deployed post-quantum algorithm(s) via Secure Software Download (SSD) or other mechanisms.

What we found in our research is that we could use the same algorithm agility feature that is built into modern PKIs to support the use of multiple keys for every aspect of the PKI lifecycle; from certificates to revocation lists, everything supports our new paradigm. In other words, our work enables the use of classic algorithms like RSA or ECDSA alongside new ones. This allows for a gradual transition to new algorithms without losing backward compatibility for devices that cannot be updated with the new algorithms. Figure 1 provides an intuitive representation of the composite crypto when used in X.509 certificates.

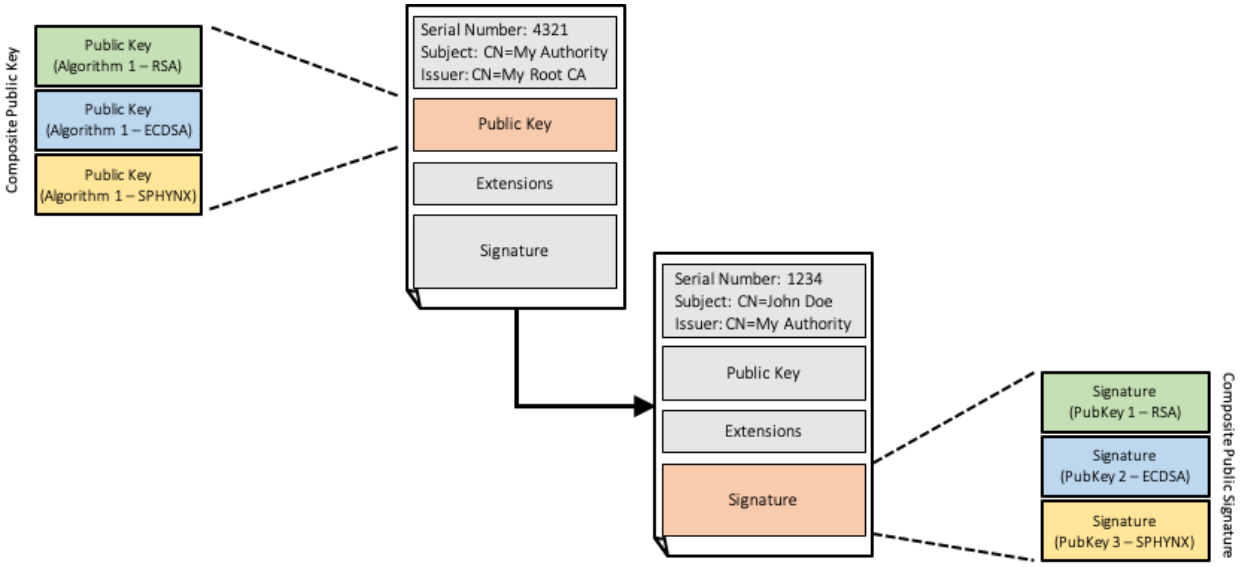

Figure 1 - Example of Composite Crypto Usage in X.509 Certificates

It is interesting to notice how our invention can be used at any time when there is either (a) the need to transition to a new protocol without establishing a completely separated infrastructure or (b) uncertainty related to the security of an algorithm over an extended period of time. For example, the solution described in this paper could be used to transition from less efficient cryptosystems (e.g., RSA) to more efficient ones (e.g., ECDSA) or to provide signatures with different hashing algorithms.

3.1. A New Paradigm: CompositeKeys and CompositeSignatures

As suggested earlier, to address our problem we relied on our initial considerations about algorithm agility to provide a simple and backward-compatible solution. In particular, we defined a new algorithm identifier and associated encoding that uses standard substructures to encapsulate multiple keys or multiple signatures in PKI data structures and authentication data.

The new type of public keys and signatures—namely, `CompositeKeys` and `CompositeSignatures`, respectively—provide the building blocks we were looking for: backward-compatible encoding for both signatures and public keys that allows for multiple keys and algorithms to be used to secure X.509 objects and produce generic signatures. By mixing classic and post-quantum algorithms, not only can all aspects of a PKI be protected today, but clients supporting at least one of the combined algorithms will be able to operate in the environment.

For example, to trust the authentication of data, relying parties might decide to verify one, some or all of the signatures depending on the ability of the relying party to support any of the algorithms used for keys and signatures.

3.1.1. Composed Public Keys and X.509 Certificates

The technical aspects of our work are simple. We first defined a new value for the `algorithm` field within the `AlgorithmIdentifier` used in the `SubjectPublicKeyInfo` of a `tbsCertificate` structure of a X.509 certificate:

```
compositeKeys OBJECT IDENTIFIER ::= {iso(1) identified-organization(3) dod(6)
                        internet(1) private(4) enterprise(1) OpenCA(18227) 10 }
```

When this value is used for the algorithm identifier, it means that the value encoded in the associated public key field (e.g., the `subjectPublicKey` field) contains multiple public keys and associated parameters. The parameters field of the `AlgorithmIdentifier` itself shall be set to `NULL` in this case as there are no specific parameters associated with the composite key – the recursive nature of the data structure allows us to delegate the parameters to the individual keys definitions.

To encode the different keys, we use a sequence of `SubjectPublicKeyInfo` objects. Each of these objects encodes the specific algorithm identifier for the specific key with its parameters and value. The final sequence is then encoded as the DER representation of the sequence of keys.

We also define the `CompositePublicKeyInfo` as a `SEQUENCE OF SubjectPublicKeyInfo` where each `SubjectPublicKeyInfo` carries the information of one public key. The ASN.1 definition of the `CompositePublicKeyInfo` is as follows:

```
CompositeSubjectPublicKeyInfo ::=  SEQUENCE (1..MAX) OF SubjectPublicKeyInfo
```

where the `SubjectPublicKeyInfo` within the `CompositeSubjectPublicKeyInfo` must not use `compositeKeys` as an algorithm identifier to prevent multiple levels of recursion.

For example, to add two separate public keys in an X.509 certificate via composite crypto, the encoding would be as follows:

```
aCompositeSubjectPublicKeyInfo  = SEQUENCE { keyInfoOne, keyInfoTwo };
  -- The main structure, a sequence of two subjectPublicKeyInfo

keyInfoOne.algorithm.algorithm  = rsaEncryption;
keyInfoOne.algorithm.parameters = NULL;
keyInfoOne.subjectPublicKey     = RSAPublicKey;
  -- The keyInfoOne provides the definition for the first key (RSA)

keyInfoTwo.algorithm.algorithm  = id-ecPublicKey;
keyInfoTwo.algorithm.parameters = EcpkParameters;
```



```
keyInfoTwo.subjectPublicKey     = ECPoint;
  -- The keyInfoTwo provides the definition for the second key (ECDSA)

aCertificate.tbsCertificate.subjectPublicKeyInfo.algorithm.algorithm = compositeKey;
aCertificate.tbsCertificate.subjectPublicKeyInfo.algorithm.params = NULL;
aCertificate.tbsCertificate.subjectPublicKeyInfo.subjectPublicKey =
                                                    DER(aCompositeSubjectPublicKeyInfo);
```

where `aCompositeSubjectPublicKeyInfo` is the sequence of two `subjectPublicKeyInfo` (i.e., `keyInfoOne` and `keyInfoTwo`). The DER representation of the `aCompositeSubjectPublicKey-Info` is then stored in the `subjectPublicKey` field of the `subjectPublicKeyInfo` of the `tbsCertificate`.

3.1.2. Composite Signatures and X.509 Certificates

When it comes to signatures in X.509 certificates and their validation, we used a similar approach. We first defined a new algorithm identifier for `compositeSignatures` and then defined the specific data structures for the composite algorithm.

Specifically, when a `compositeSignatures` schema is used to encode multiple signatures at once, the value for the algorithm identifier associated with the signature is defined as follows:

```
compositeSignatures OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
                dod(6) internet(1) private(4) enterprise(1) OpenCA(18227) 11 }
```

When the `compositeSignatures` identifier is used, the corresponding value encoded in the `signatureValue` field contains multiple signatures and associated parameters encoded as the DER representation of a `CompositeSignatureValue` that is a `SEQUENCE OF SignatureInfo`. Each `SignatureInfo` carries the information about one of the signatures applied to the certificate. The definition of the `CompositeSignaturesValue` is as follows:

```
  CompositeSignaturesValue  ::=  SEQUENCE (1..MAX) OF CompositeSignatureInfo
```

For example, to encode signatures made with two separate keys (one RSA key and one EC key), the encoding would be as follows:

```
aCompositeSignatureInfo         = { sigInfoOne, sigInfoTwo };
  -- The main structure, a sequence of two SignatureInfo

sigInfoOne.algorithm.algorithm  = rsaEncryption;
sigInfoOne.algorithm.parameters = NULL;
sigInfoOne.subjectPublicKey     = <RSA Signature Value>;
  -- The sigInfoOne provides the definition for the first signature (RSA)

sigInfoTwo.algorithm.algorithm  = id-ecPublicKey;
sigInfoTwo.algorithm.parameters = EcpkParameters;
sigInfoTwo.subjectPublicKey     = <ECDSA Signature Value>;
  -- The sigInfoTwo provides the definition for the second signature (ECDSA)

aCertificate.signatureAlgorithm.algorithm.algorithm = compositeSignatures;
aCertificate.signatureAlgorithm.algorithm.params    = NULL;
aCertificate.signatureValue = DER(aCompositeSignatureInfo);
  -- The final encoding of multiple signatures in a certificate
```

where the `aCompositeSignatureInfo` structure contains the sequence of the two `SignatureInfo` (i.e., `sigInfoOne` and `sigInfoTwo`). The DER representation of the `aCompositeSignatureInfo` is then used for the `signatureValue` field of the certificate structure.

3.1.3. Generating Composite Signatures

To generate composite signatures, the signer shall generate each signature independently by using each of the keys present in the signer's `CompositePublicKeyInfo` in the same order they appear. Specifically, the signer shall use the first key to generate the first signature, the second key to generate the second signature, and so on. The signer shall generate one signature for each key in the key set.

For example, if the `CompositeSubjectKeyInfo` has three public keys (K_1, K_2 and K_3) of types RSA, EC and DSA, respectively, the signing party shall generate the first signature by using K_1, the second signature by using K_2, and the last signature by using K_3.

3.1.4. Verifying Composite Signatures and Time-Dependent Validation Policies Deployment

To be able to verify composite signatures, a relying party shall verify each of the applied signatures independently. Also in this case, the relying party shall verify the signature by using the corresponding public key in the signer's certificate in order—that is, the order of the signatures within the `CompositeSignature` shall respect the order of the keys in the `CompositePublicKeyInfo` in the certificate.

For example, if the certificate has a `CompositeSubjectPublicKeyInfo` that contains three keys (K_1, K_2 and K_3) of types RSA, EC and DSA, respectively, the relying party shall verify the first signature in the composite signature by using K_1, the second signature by using K_2, and the last signature by using K_3.

One important aspect of our invention is that it can be combined with the possibility of applying validation policies that can be changed over time or remain static.

In a static configuration, for example, the relying party might set its policy not to evaluate the correctness of signatures if they do not support any of the used (or specific) algorithms, or otherwise refuse to trust the signed data entirely, even if it is not able to verify just one of the composite signature's elements.

In a time-dependent policy, relying parties could instead use the quantum threat risk level to set the threshold for the policy change. Imagine, for example, an infrastructure in which both RSA and ECDSA algorithms are used via `CompositeKeys` encoded in the root and intermediate CA certificates. Also imagine that the RSA algorithm is set to retire because deemed not secure in 10 years (e.g., the used key sizes for the infrastructure are not considered secure anymore). In addition, assume that ECDSA will still be considered secure for the application (e.g., larger keys can be deployed here because of better performances). A validation policy could allow relying parties to validate composite signatures by using only the RSA algorithm for the next 10 years. After that, the policy might mandate relying parties to validate signatures by using all algorithms to make sure the stronger one(s) is validated too.

In the post-quantum scenario, this translates to a very similar approach.

Specifically, although static policies might be more appropriate for those relying parties or devices whose crypto cannot be updated (see the next section), more dynamic ones could be deployed when the validation of new algorithms can be added to the device. In this case, up to a certain security risk level

(e.g., until practical deployment of quantum computers is achieved), relying parties and devices could still be allowed to use just the traditional algorithms for validation and enable the new ones when the risk level for the involved stakeholders goes over the acceptable threshold (and support for it is successfully deployed).

3.1.5. Use of Composite Crypto for Backward Compatibility

The same solution can be used when deploying a new infrastructure where participants in the ecosystems might not be able to update their security parameters. In this case, composite crypto structures can be used to deploy trust infrastructures where new and old algorithms coexist in the `CompositeKeys` and `CompositeSignatures` of certificates. The deployment of superseded algorithms along with new ones allows relying parties that cannot update their cryptographic suites (e.g., devices that are already in the field) to participate in the same infrastructure while still allowing other relying parties to use stronger validation algorithms.

In other words, composite crypto can be used to keep using old algorithms to accommodate for older, already-in-the-field devices that might have hardware constraints (e.g., they have a secure element that cannot be replaced) without compromising the overall security of the infrastructure. The stronger keys/algorithms will be used by more capable devices (e.g., P-521 or quantum-resistant algorithms).

4. A Complete Solution: From Requests to Revocation

Although other solutions have been tried to provide support for adding multiple keys or algorithms to certificates by adding new types of extensions, no other solution actually tackles all the aspects of the PKI lifecycle. Specifically, no other solution is available that addresses not only the authentication of certificates but also the authentication of certificate requests and revocation objects. In this section, we take a look at the applicability of our solution to these aspects that are central to the correct behavior of PKIs.

4.1. Requesting Composite Certificates

In PKIs, the [PKCS10] standard is commonly used when it comes to requesting certificates. Many standard (and non-standard) protocols use it as the core building block for their own certificate request messages. Examples of this can be found in Certificate Management over CMS (CMC) [RFC 5272, RFC 5273] and in the Automated Certificate Management Environment (ACME) [RFC 8555].

Fortunately, our work is compatible with the PKCS#10 format.

In particular, to authenticate PKCS#10 requests with composite crypto, the `signatureAlgorithm`'s algorithm identifier in the `CertificationRequest` structure can be set to carry the `compositeSignatures` value, and the `parameters` one can be set to `NULL`.

The `signature` field is the one that carries the DER representation of `CompositeSignatures` and contains all the signatures generated with the `compositeKeys` associated with the identity that is requesting a certificate. The signatures are calculated, as usual, over the DER representation of the `certificationRequestInfo` field of the `CertificationRequest`.

4.2. Use of Composite Signatures in CRLs

CRLs are the oldest form of revocation for X.509 certificates [RFC 5280, RFC 5759, RFC 6818]. Their structure was inspired by credit-card number blacklists and are used to convey the list of serial numbers of certificates that have been revoked (together with an optional reason code). Because this list is often signed by the issuing CA (or a designated signer), we need to make sure that this list is securely authenticated, even in a post-quantum threat scenario.

Our approach seamlessly works with CRLs too.

As with the case of X.509 certificates, the `signatureAlgorithm` field in the `CertificateList` structure can be set to carry the `compositeSignatures` value, and the `parameters` field can be set to `NULL`. The `signature` field of the `CertificateList` can be set to carry the DER representation of the `CompositeSignaturesValue`.

Also, in this case, there is no change in how the signatures are generated because the individual signatures are calculated over the DER representation of the `tbsCertList`, as usual.

4.3. Use of Composite Signatures in OCSP Requests and Responses

OCSP requests and responses have signature fields that can be leveraged with composite signatures to address the quantum threat without requiring any protocol changes.

To authenticate OCSP requests, the `signatureAlgorithm` algorithm identifier in the `Signature` structure of the `OCSPRequest` can be set to `compositeSignatures`, and the `parameters` field can be set to `NULL`. The corresponding `signature` field of the `Signature` structure can then hold the DER representation of the `CompositeSignature` value itself. The signatures are calculated, as usual, over the DER representation of the `tbsRequest` in the `OCSPRequest` structure.

For OCSP responses, the `BasicOCSPResponse` structure provides, together with the `tbsResponseData` and the `signatureAlgorithm` ones, the needed fields to host composite signatures. Specifically, the `signatureAlgorithm` algorithm identifier in the `BasicOCSPResponse` structure can be set to `compositeSignatures`, and the `parameters` field can be set to `NULL`. The corresponding `signature` field can be set to hold DER representation of the `CompositeSignaturesValue`. Also in this case, the individual signatures can be calculated and encoded, as usual, over the DER representation of the `tbsResponseData` field of the `BasicOCSPResponse`.

5. Composite Cryptography and Hardware Integration

Modern cryptography relies on two main principles: making algorithms public and moving all security properties to the secrecy of keys. That is why one of the pillars of modern cryptography is keeping your secrets ... secret! This is true not only for shared secrets but also for private keys.

To help with the security of keys, best practices commonly require the use of HSMs or secure elements in our devices to make sure that (a) private key computations safely happen on a dedicated processor and (b) their value cannot be extracted by a remote party.

From this point of view, the quantum threat not only poses a risk from a security perspective, but it also requires hardware updates to provide the same security properties that we have enjoyed for decades.

Although quantum-resistant algorithms must be run in software until support for them is provided via secure hardware implementations, the composite cryptography solution itself is fully compatible with existing hardware and security modules. This is because the processing of the individual keys and signatures still relies on the same primitives; therefore, no changes are required for composite crypto integration, as long as the algorithm is supported by the crypto accelerator. Similarily, current standard interfaces to crypto hardware, like PKCS#11 [PKCS11] and supporting libraries, do not require any changes for the same reasons. This is extremely important for preserving backward compatibility with deployed HSMs, which are usually large and very expensive pieces of equipment used mostly to secure operating CA keys.

This means that all investments that Certificate Service Providers (CSPs) made in purchasing and maintaining their HSMs are not impacted, as no changes are needed to leverage composite crypto. For example, existing root and intermediate CAs can add new keys to their own certificate and have their new request signed by using composite crypto today. This allows current infrastructures to add new algorithms and still be able to leverage the security (and certifications) of today's crypto hardware (e.g., FIPS 140-2).

Summarizing, composite crypto can be used today with existing certified hardware components, thus allowing the transition from, for example, RSA to ECDSA without the need for new hardware or certifications. Support for new algorithms is still required for deploying quantum-safe crypto.

6. Deploying a Backward-Compatible, Quantum-Safe DOCSIS PKI

So far, the DOCSIS ecosystem has deployed two different infrastructures throughout its lifetime. The first one, today referred to as "legacy PKI," was deployed 20 years ago and provides its services for DOCSIS 1.1–3.0 devices. Because this infrastructure is set to expire soon, it was reasonable to focus our efforts on the newer infrastructure.

The second deployed infrastructure, or "new PKI," was introduced to update the security parameters for the whole broadband industry and provides its services to secure not only DOCSIS 3.1–4.0 devices but also other entities associated with the existing and upcoming distributed architectures (e.g., R-PHY or CCap Core). Because this infrastructure is not going to sunset anytime soon, our work has been focused on defining the deployment strategy for securing this second "new" infrastructure across the "quantum-threat hump."

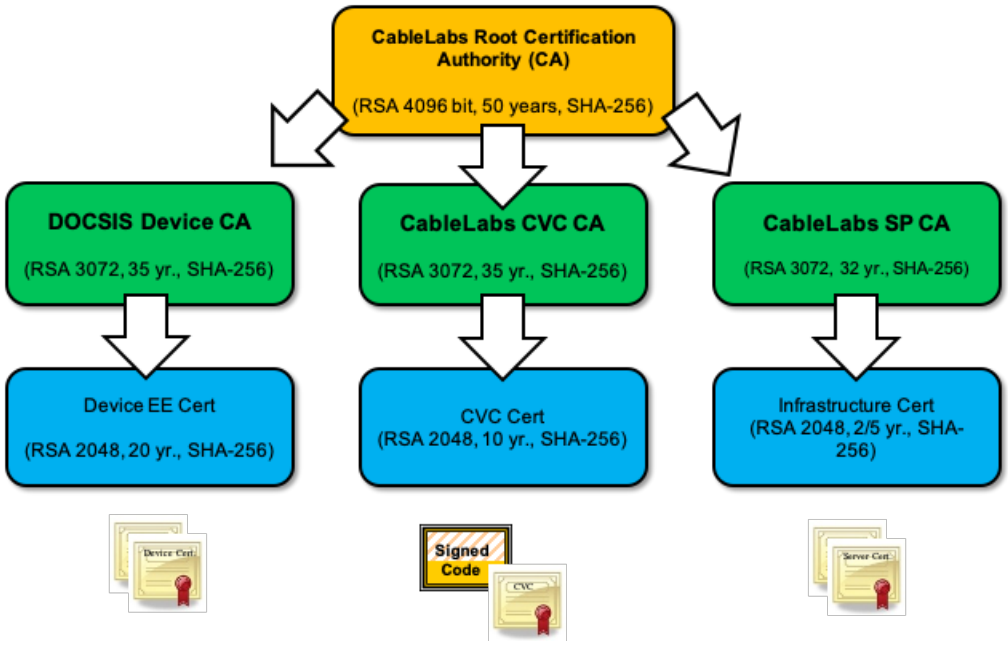

Figure 2 - The DOCSIS "New PKI" Hierarchy

Figure 2 depicts the "new PKI" structure that comprises a three-tier hierarchy with an offline root CA that issues a second level of intermediate CAs. These CAs issue only end-entity certificates and have assigned operational scopes (e.g., device certificates vs. code-signing certificates) that limit their liability in case of compromise. All participating entities in the infrastructure use the RSA algorithm for their keys.

Currently, our work is focused on setting up a test infrastructure that uses composite keys and signatures to secure the core part of the infrastructure (i.e., the root CA and the intermediate CAs) against the quantum threat and, at the same time, provide the possibility to leverage algorithms other than RSA for increased efficiency.

The envisioned test infrastructure mimics the current "new PKI" hierarchy and uses, for the core of the hierarchy (i.e., root and intermediate CAs), three separate public keys: the current RSA key, a new ECDSA key and a new Post-Quantum-Algorithm (PQA) one. Device or end-entity certificates are issued with a single algorithm that can be either RSA, ECDSA or the PQA, depending on the entity or device capabilities. Network or server-side identities are issued with composite crypto keys comprising all the deployed algorithms to support all classes of devices (i.e., classic and post-quantum).

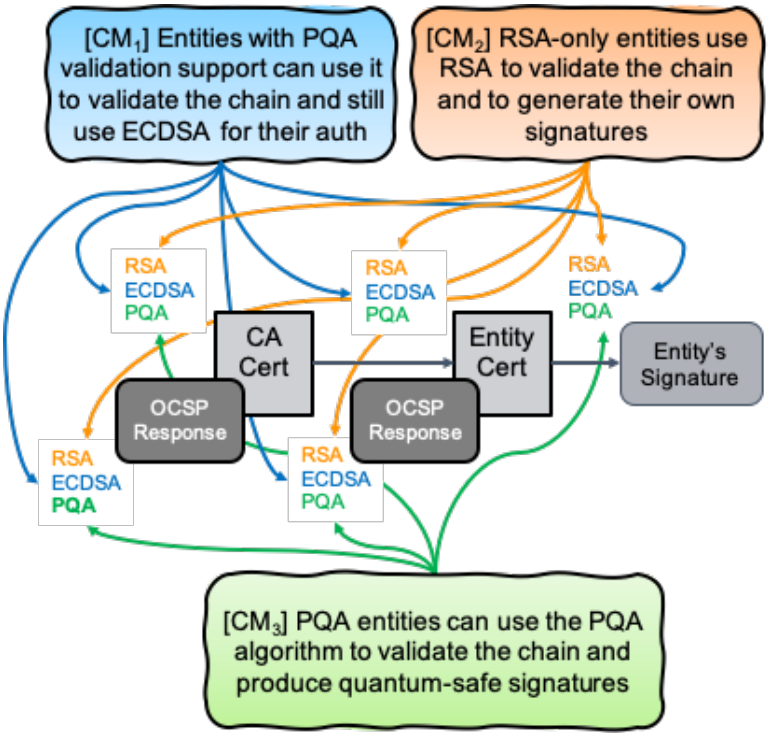

Figure 3 - Validation Scenario Showing Multiple Entities with Different Capabilities

Figure 3 provides a validation scenario in which three devices with different capabilities are authenticating themselves with credentials from the quantum-safe PKI. In this scenario, CM_1 is capable of only working with ECDSA P-256 keys but can validate PQA signatures and public keys, CM_2 is capable of only working with RSA keys, and CM_3 fully supports PQA (not only validation but also signing and key management) and can also validate RSA and ECDSA.

On the client side, CM_1 can benefit from using either classic or quantum-safe crypto and still authenticate itself by using classic crypto (ECDSA). CM_2, instead, is a constrained device and can only use RSA; without any further update, this class of devices cannot be securely authenticated under the quantum-threat model (see the next section for proposed mitigations to address this use case). CM_3 is a newer device that fully supports the selected PQA and therefore uses that algorithm to both validate the network credentials and generate its own authentication traces. In case CM_3 also supports classic cryptography, RSA or ECDSA validation algorithms can still be used before quantum-based attacks become practical.

On the network side, the CMTS must support all the algorithms supported by the entire population of deployed devices—both for validation and authentication. To be able to authenticate CM_1, the CMTS must support ECDSA. To be able to authenticate CM_2, instead, the CMTS has to support RSA. Ultimately, authenticating CM_3 requires the CMTS to support the PQA. On the authentication side, to allow devices that might support only one of the deployed algorithms to be able validate the network credentials (e.g., in Baseline Privacy Plus Interface [BPI+] V2), the CMTS must support all of the used algorithms in its composite key. This allows the use of RSA, ECDSA and PQA on the various classes of devices without the need of separate identities or infrastructures.

In the rest of this section, we provide an overview of the proposed approaches for tackling the quantum threat when deploying full PQA-based solutions is not an option—for example, because of limitations in fielded devices.

6.1. Deploying Post-Quantum Solutions for RSA-Only Capable Devices

One of the most challenging issues when it comes to cryptography is to include devices with different capabilities, and some of these devices might not be upgradeable. This can be due to software limitations (e.g., firmware or applications cannot be securely updated) or hardware limitations (e.g., crypto accelerators or secure elements).

Although composite cryptography cannot be used to secure non-quantum-safe authentications against the quantum threat by itself, in case entities and devices do not to support any post-quantum algorithm, we identified a solution that can be used to extend the lifetime of deployed devices for the broadband industry.

6.1.1. Fielded Devices and Authentications

To consider securing fielded devices that cannot be updated to support PQAs, we looked at their current capabilities. We researched which classes of quantum-safe algorithms are available and how can we leverage them, given today's hardware constraints, to provide quantum-safe authentications.

What we found is that especially for constrained devices, the only option at our disposal is the use of pre-shared keys (PSKs) to allow for post-quantum safe authentications for the various identified use cases. We looked at the limitations and how to tackle them when planning for the transitioning. In this scenario, the post-quantum PSK (PQP) is used to generate quantum-safe signatures and, in some cases, to also provide a "second factor" of authentication for the certificate chain when no PQA support is available.

To generate quantum-safe signatures, devices can start using the PSK with their device private keys (e.g., RSA keys) to produce quantum-safe authentication data; the classic cryptography provides the identity information, along with the proof of possession of the classic private key, while the PQP provides the security of the message via a symmetric signature. Combining the PSK with the authentication process can be done in different ways. A hash-based key derivation function (HKDF) [RFC 5869] can be used with the PQP to derive a message-specific key that is then used with an HMAC function to authenticate the messages.

In BPI+ Version 2[1], because of the use of the Cryptographic Message Syntax (CMS) [RFC 5652], combining PSKs with DOCSIS authentication could also be used to provide key encapsulation capabilities for delivering authorization keys via a quantum-safe mechanism [RFC 8696]. For previous versions of DOCSIS, or where BPI+ V2 is not supported because direct RSA encryption of the authorization key is used, additional changes to the protocol messages might also be required.

Let's now take a look at two different scenarios and how to address their limitations. The first use case assumes that private keys, certificates and crypto capabilities cannot be updated—in this case, we rely on traditional crypto to perform the needed setup operations securely before the quantum threat is real. To remove the requirement for time-safe deployment, a second proposal is introduced that looks at devices whose private keys cannot be updated, but their support for composite crypto and quantum-safe KEX algorithm can (e.g., via SSD).

[1] The new version of BPI+ was recently introduced in the DOCSIS 4.0 specifications to introduce several improvements to the authorization process, such as mutual authentication and perfect forward secrecy.

6.1.1.1. Immutable Devices and Quantum-Safe Traditional Authentications

In this scenario, we look at entities and devices whose support for new algorithms cannot be updated—not even for validation-only operations (i.e., no support for private keys or signing). Because of these restrictions, our proposal is to leverage traditional cryptography to distribute per-device PSKs that can be leveraged for post-quantum authentications.

Specifically, our proposal is to enhance the DOCSIS protocol to introduce the possibility to securely transfer (or derive) a common PSK between the operator's network and the device being authenticated (i.e., the cable modem or the R-PHY node). Once the PQP is securely delivered to the device, this secret can stay dormant until needed for generating quantum-safe signatures. This PSK can be deployed as part of the initial registration of devices to the network, or it could be initiated at any time as long as the PQP is transferred securely.

One very important aspect of the solution is to make sure not only that the session parameters are properly authenticated via the quantum-safe signature, but that the certificate chain is protected against modifications by including it into the original signature. Assuming the PSK is secure, the relying party (e.g., the CCap Core or the CMTS) can trust both the signature and identity of the device because of the security of the PQP.

The big limitation here is related to the security of the PSK. Because the PSK has to be transferred or derived by using traditional cryptography, an attacker could potentially pre-record the device's traffic and then—when access to a quantum computer is obtained—get access to the PQP by breaking the classic KEX algorithm. The attacker would then be able to impersonate any device, even when using the PQP when generating signatures. Although a more secure solution is provided in the next section, operators can make things difficult for a malicious attacker who is pre-recording DOCSIS traffic to analyze and decrypt it at a later time.

Indeed, operators can deploy keys at random intervals or use procedures for combining new and old values (and/or replace them) at random times. An attacker would require knowledge of the whole history of the device connectivity to be able to attack its PQP.

6.1.1.2. Partially Upgradeable Devices and Quantum-Safe Traditional Authentications

When entities and devices can be updated to support new algorithms, but their private keys cannot (e.g., they are tied to secure elements that cannot be updated), more secure options can be adopted for transferring the PSK. We think that this upgrade path might be the most common for the broadband industry given that the possibility to provide secure software updates is built into the DOCSIS protocol since its inception.

In this scenario, we assume that devices have been updated to support composite crypto and a quantum-safe KEX algorithm, but they cannot update their own private keys. We also assume that the root and the intermediate CAs have been deployed with a PQA algorithm alongside traditional ones.

To protect the PQP against a possible all-powerful adversary that can break the traditional cryptography, we share the PQP by using a quantum-safe KEX algorithm. The use of a quantum-safe KEX algorithm guarantees that an adversary would not be able to have access to the PQP, even when pre-recording encrypted sessions. The use of traditional cryptography still provides the needed secure identity validation to make sure the PQP is shared across the right entities. Also, in this case, when generating authentication

data, we need to protect the certificate chain because the link between the intermediate CAs and the end-entity certificate is not yet protected via a PQA.

When the entity's certificate can be updated and signed by using a PQA (i.e., the CA signs a new certificate for the entity that includes the original "traditional" key of the entity only, and it is signed using all the keys in the CA's composite certificate), the need to sign the certificate chain with the PSK can be relaxed. In fact, because the links from the root to the end entity is already secured by the use of a PQA, no additional use of the PSK to protect the device's or CAs' identities is needed.

7. Conclusion

In this paper, we have examined the quantum threat, the current status of post-quantum cryptography and the associated standardization efforts. We also describe how the quantum threat will affect the various classes of algorithms we use today within the broadband industry.

The core of this paper provided a description of our novel approach based on composite cryptography, and how it can be used to address the quantum threat. Specifically, we provided the technical description of the composite cryptography building blocks (`CompositeKeys` and `CompositeSignatures`) and showed how to integrate them in every aspect of modern trust infrastructures and associated services (e.g., certificates, CRLs, OCSP). We then explored our proposal for a quantum-resistant and backward-compatible DOCSIS PKI and the use of PSKs to secure entities and devices that will not have access to quantum-resistant cryptography.

Our future efforts will be aimed at working on open source tools and test environments. The quantum-safe test services deployment will be paramount for security experts and researchers to experiment with combining different quantum-resistant algorithms and study their interactions with our protocols. The selection of the protocols and their parameters will pave the road to quantum-resistant DOCSIS implementations and deployment.

Ultimately, the takeaway message from our work is that the quantum threat is closer that many people think, and we need to be already preparing our infrastructures and protocols for the upcoming revolution. Not only quantum computers will become more capable of handling more complex problems; the advancements in quantum-based algorithms put everybody's security and privacy at risk.

Although the deadline for planning to address this new class of threats is fast approaching, our work shows how the broadband industry can already start to address them today and lead the transition to quantum-safe cryptography to be able to continue to protect users' privacy and securely deliver top-quality services.

Abbreviations

AES	Advanced Encryption Standard
BPI+	Baseline Privacy Plus Interface
CA	certification authority
CCAP	Converged Cable Access Platform
CRL	certificate revocation list
CSP	certificate service provider
DER	Distinguished Encoding Rules
DOCSIS	Data Over Cable Service Interface Specifications
EC	Elliptic-Curves
ECC	Elliptic-Curves Cryptography
ECDH	Elliptic-Curves Diffie-Hellman
ECDSA	Elliptic-Curves Digital Signing Algorithm
EE	end entity
DH	Diffie-Hellman
HSM	hardware security module
IETF	Internet Engineering Task Force Standards Organization
ISBE	International Society of Broadband Experts
KEM	key encapsulation mechanism
KEX	key exchange (algorithm)
MLWE	module learning with errors
MLWR	module learning with rounding
MSIS	module short integer solutions
NIST	National Institute of Standards and Technologies
NTT	Number Theoretic Transform
PKC	public-key cryptography
PKCS#10	Public Key Cryptography Standard 10 (certificate request)
PKCS#11	Public Key Cryptography Standard 11 (hardware interface)
PKI	public-key infrastructure
OCSP	Online Certificate Status Protocol
PFS	perfect forward secrecy
PQA	post-quantum algorithm
QC	quantum computing
R-PHY	Remote RF Layer (PHY)
R-MACPHY	Remote Media Access Control and RF Layer (PHY)
RSA	Rivest-Shamir-Adleman (cryptosystem)
SHA-256	Secure Hash Algorithm 2 (256-bit)
SHA-3	Secure Hash Algorithm 3
SCTE	Society of Cable Telecommunications Engineers
SE	secure element
SSD	secure software download
TLS	Transport Layer Security
UOV	unbalanced olive-vinegar (construct)

Bibliography & References

Doc40: *Data-Over-Cable Service Interface Specifications, DOCSIS 4.0, Security Specifications.* CableLabs Publication, 2019. Available as CM-SP-SECv4.0-IO1-190815.

Doc31: *Data-Over-Cable Service Interface Specifications, DOCSIS 3.1, Security Specifications.* CableLabs Publication, 2020. Available as CM-SP-SECv3.1-IO9-200407.

Ntru9: *Institute of Electrical and Electronics Engineers (2009) IEEE Standard1363.1-2008 – Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices* (IEEE, Piscataway, New Jersey, United States). Available at https://doi.org/10.1109/IEEESTD.2009.4800404

Ntru10: *American National Standards Institute (2010) ANSI X9.98-2010 – Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry* (ANSI, New York City, United States). Available at https://webstore.ansi.org/standards/ascx9/ansix9982010r2017

Itu509: ITU-T Recommendation X.509 (2005) | ISO/IEC 9594-8:2005, *Information Technology - Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.

Rphy18: *Data-Over-Cable Service Interface Specifications, DCA – MHAv2*. Remote PHY Specification. Available as CM-SP-R-PHY-I10-180509.

Com20: M. Pala. *Composite Public Keys and Signatures,* IETF I-D 03. July 2018.

GE19: C. Gidney and Martin Ekerå, *How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*, Available at https://arxiv.org/abs/1905.09749

Nist20: National Institute of Standards and Technology, Post-Quantum Cryptography – Round 3 Submissions. Available at https://csrc.nist.gov/Projects/post-quantum-cryptography

NISTIR: Gorjan Alagic et Al., Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process, NIST, July 2020. Available At https://csrc.nist.gov/publications/detail/nistir/8309/final

Gro96: Lov K. Grover. *A Fast Quantum Mechanical Algorithm for Database Search.* In Gary L. Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22–24, 1996, pages 212–219. ACM, 1996.

Dr99: Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

Aes16: Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Stein-wandt. *Applying Grover's Algorithm to AES: Quantum Resource Estimates.* In PQCrypto, Volume 9606 of Lecture Notes in Computer Science, pages 29–43. Springer, 2016.

Aes20: Xavier Bonnetain, María Naya-Plasencia and André Schrottenloher. *Quantum Security Analysis of AES. IACR Transactions on Symmetric Cryptology* Vol. 0, No. 0, pp.1–3, 2020.

RFC 8696: IETF RFC 8696, R. Housley, *Using Pre-Shared Key (PSK) in the Cryptographic Message Syntax (CMS),* December 2019.

RFC 8555: IETF RFC 8555, R. Barnes, et al., *Automatic Certificate Management Environment (ACME),* March 2019.

RFC 8446: IETF RFC 8446, E. Rescorla, et al., *The Transport Layer Security (TLS) Protocol, Version 1.3,* August 2018.

RFC 6818: IETF RFC 6818, P. Yee, *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,* January 2013.

RFC 6402: IETF RFC 6402, J. Schaad, *Certificate Management over CMS (CMC) Updates,* November 2011.

RFC 5869: IETF RFC 5869, H. Krawczyk and P. Eronen, *HMAC-based Extract-and-Expand Key Derivation Function (HKDF),* May 2010.

RFC 5758: IETF RFC 5758, Q. Dang, et al., *Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA,* January 2010.

RFC 5652: IETF RFC 5652, R. Housley, *Cryptographic Message Syntax (CMS)*, September 2009.

RFC 5280: IETF RFC 5280, W. Polk, et al., *Cryptographic Message Syntax (CMS),* May 2008.

RFC 5273: IETF RFC 5273, J. Schaad, et al., *Certificate Management over CMS (CMC): Transport Protocols*, June 2008.

RFC 5272: IETF RFC 5272, J. Schaad, et al., *Certificate Management over CMS (CMC),* June 2008.

RFC 3279: IETF RFC 3279, W. Polk, et al., *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, April 2002.

RFC 2986: IETF 2986, M. Nystrom, et al., *PKCS #10: Certification Request Syntax Specification*, Version 1.7, November 2000.

PKCS11: OASIS Standard, S. Gleeson and C. Zimman, *PKCS #11 Cryptographic Token Interface Base Specification,* Version 2.40, April 2015.

