SCTE CABLE-TEC EXPO FAST FORWARD 2021
VIRTUAL EXPERIENCE
OCTOBER 11-14

2021 Fall Technical Forum
SCTE • NCTA • CABLELABS

Enabling Encryption and Algorithm Revocation for Post-Quantum DOCSIS Certificates

Novel Results in Multi-Key Trust Environments Deployments

A Technical Paper prepared for SCTE by

Dr. Massimiliano Pala

PKI Architectures Team, Director

Cable Television Laboratories, Inc.

858 Coal Creek Cir, Louisville, CO

603.369.9332

m.pala@cablelabs.com

Table of Contents

List of Figures

List of Tables

List of Equations

1. Introduction

The cryptography world is going through a revolution. As new computation paradigms emerge and rapidly advance, like quantum computing (QC), the broadband industry needs to start planning how it will address the new security threats that are on the horizon.

Most of the public key cryptosystems like RSA [Rsa16] or ECDSA [Ec05] will not be considered secure when (and if) a large quantum supercomputer is ever built. For the broadband industry this means that, because of the dependency on X.509 [X509] certificates and the RSA algorithm, to provide devices with secure and verifiable identities, the protocols that are used today, e.g. DOCSIS® protocols [Doc31; Doc40], will need to support new algorithms and identities. In fact, network elements like cable modems or Remote PHY (R-PHY) nodes [RPhy18] use, today, their RSA private key and associated certificates chain to prove they are a legitimate and registered entity on the network. To continue to benefit from the security and usability advantages of public-key cryptography (PKC), the broadband industry must provide a mechanism for transitioning to quantum-resistant solutions *in a cost-effective manner*. Although our previous results on Composite Crypto (or Hybrid certificates) provided a promising path forward for the deployment of multiple keys associated with a single identity, our work still left some important questions. For example, an area that was still left to be explored was how to handle complex crypto policies for algorithm validation and deprecation. Because of these limitations, encryption was also left out of scope.

This paper describes our new results in multi-key environments that address the open issues from our previous work and update its technical details [Pala04]. Specifically, in this work we extend the initial proposal and introduce the explicit separation of “AND” and “OR” logic operations across the multi-key signature components. Additionally, our work enables encryption for multi-key certificates (e.g., for S/MIME or document multi-signing purposes) that was, up to now, still an open problem. Together with these important results, this paper also describes our proposal for algorithm revocation and how we leverage the details of X.509 certificates’ public key structures together with extensions in CRLs and/or OCSP responses to provide a dynamic, centrally managed, and easy to deploy algorithm revocation mechanism.

The rest of the paper is organized as follows: Section 2 provides an overview of the current landscape of Post-Quantum (PQ) cryptography and how it addresses the quantum threat. Section 3 describes the composite crypto solution and highlights current limitations of multi-key certificates when it comes to validations or encryption; Section 4 describes the new results that stem from the introduction of Combined Crypto alongside Composite Crypto; Section 5 provides the details on our algorithm revocation mechanism. Section 6 addresses the multi-key encryption conundrum and, finally, Section 7 provides our conclusions and envisioned future work.

2. The Post-Quantum Cryptography Landscape

Although the standardization process that is currently undergoing at NIST has not yet completed, there are interesting trends and practical long-term considerations for PQ algorithms deployment within the broadband industry that we can already highlight.

In order to understand how the new algorithms address quantum resistance, it is important to look at the principles behind solving the Hidden Subgroup Problem (or HSP) and how quantum computers can leverage superposition, entanglement, and interference to efficiently solve HSP for relevant domains.

2.1. The Hidden Subgroup Problem (HSP) and Factoring Keys

When it comes to the link between "classic" cryptography, like RSA or ECDSA, and quantum-based factorization algorithms, such as the one proposed by Schorr, it is not always easy to understand how periodicity comes into play and how post-quantum algorithms address quantum-resistance.

In this section we provide a qualitative explanation of HSP for the classic and post-quantum use-cases by introducing group theory concepts and their intersection with cryptography.

2.1.1. Groups, Cosets, and H-Periodic functions

With the use of modular arithmetic, a cornerstone in modern cryptography, we introduce, de facto, periodicity in the form of cyclic groups. These mathematical constructs consist of a set (e.g., "integers mod N") and a binary operation that takes two inputs and generates outputs in the same set:

$$G \times G \longrightarrow G$$

A group is characterized by three fundamental properties: (a) associativity, (b) a neutral element, and (c) all elements in the group have an inverse. Commutativity is not a core characteristic of a group and this, specifically, is a key differentiator when looking at quantum-resistance as explained in the rest of this section. A commutative group is also called an Abelian group.

When it comes to group theory, there are two definitions that must be well understood: subgroups and cosets. A subgroup H of a group G is defined as a group that still satisfies the group properties and is generated by one or more elements of the group (e.g., "h"). A coset is a similar concept to a subgroup in the sense that it can be seen as "translated" subgroups with respect to an element of the group G. For example, given a subgroup H generated by two elements (h_1 and h_2) and a third element "x" in the group G, the "coset of H with representative x (element of G)" is generated by applying all the permutations starting from the element "x" instead of starting from the neutral element.

The definition of H-periodic functions is strictly connected to the definition of cosets. When a function maps the values of a group to a set, it is said to be H-periodic (H is a subgroup) if, for all cosets xH, the value of the function is the same on all the elements in xH and differs on all elements of the other cosets.

2.1.2. The Hidden Subgroup problem and classic cryptography

The solution to HSP over specific groups can lead to breaking classic and post-quantum cryptography by leveraging the ability of a quantum computer to efficiently find the period of the underlying H-periodic function.

An example of this approach is explained in the famous Schor's paper from 1997 [Shor97]. In his work, Shor teaches us how to use quantum computers to efficiently implement the period finding function which is at the core of the factorization problem. In the case of "classic" algorithms, like the RSA or ECDSA, the underlying groups are commutative and, therefore, easier to deal with. For example, the group definition for the RSA case is $G_{RSA}= (\mathbb{Z}_N, +)$, while $G_{ECDSA}= (\mathbb{Z}_N \times \mathbb{Z}_N, +)$ provides the definition for the ECDSA one. The commutative property of these groups allows us to use the Fourier analysis on Abelian groups by using the Quantum Fourier Transform operation. In the RSA case, for example, given access to the function f that computes exponentiation modulo n, the factorization problem can be reformulated as finding a generator of the subgroup $H = \varphi(n)\mathbb{Z} \in \mathbb{Z}$, where $\varphi(n)$ is the group order and $\mathbb{Z}/n\mathbb{Z}$ is the set of integers modulo n. We can then use the function f as the oracle function for the subgroup H as:

$$f: \mathbb{Z} \to \mathbb{Z}/n\mathbb{Z} : x \mapsto a^x mod\ n$$

Once the generator $\varphi(n)$ is found through repeated sampling, computing the factorization of n can be accomplished by using the greatest common divisor (GCD) technique to find the non-trivial factors from the measurements.

2.1.3. The Dihedral Hidden Subgroup problem and Lattices

An interesting group that is relevant for post-quantum cryptography is the symmetry group. This group is defined as the set of all the possible elements permutations π (i.e., N-1 rotations, N reflections, and the neutral element) together with the functional composition operation. The neutral element is, in this case, the permutation that maps everything to itself, i.e., the identity element.

When looking at post-quantum algorithms and their relationship with HSP, we need to start from Regev's 2002 work on HSP. In his paper on Quantum Computation and Lattice Problems [Reg02], Regev shows how a solution to the Unique Shortest Vector Problem (SVP) can be obtained under the assumption that an algorithm that solves the hidden subgroup problem on the dihedral group by coset sampling exists. Regev's work demonstrates the equivalence between solving SVP on lattices and solving HSP on the dihedral group. The main difference with the classic use case is the fact that the group (i.e., domain and operation) for which a solution of the HSP is needed are non-commutative. As a reminder, the dihedral subgroup is a subset of all the permutations that are automorphisms (or symmetries) of the N-cycle (i.e., all the permutations that preserve the structure of the N-sided regular polygon) which include rotations and reflections. The Hidden Subgroup Problem for the dihedral subgroup can be defined as "given an H-periodic function, find H (or find the generators of H)".

Although also in the noncommutative subgroup problem the use of the Fourier analysis is at the center of efficient quantum-based solution, the difficulties of performing it on noncommutative groups makes the noncommutative version of the problem very challenging. Ettinger and Høyer [HeHø04] showed that efficiently solving the HSP for noncommutative groups is possible. More precisely, they show that it is possible to obtain sufficient statistical information about the hidden subgroup with a polynomial number of queries (similarly to the "classic" use case) … However, no known efficient algorithm exists that can leverage this information to find the generator for the subgroup. In their paper, Ettinger and Høyer state:

"Our main result is that there exists a quantum algorithm that solves the dihedral subgroup problem using only a linear number of evaluations of the function which is given as input [...] However, we hasten to add that our algorithm does not run in polynomial time. […] the algorithm applies a certain quantum subroutine a linear number of times […]. We know how to find the subgroup from the data in exponential time, but we do not know if this task can be done efficiently."

The original algorithm from Kuperberg from 2003 to solve HSP on the dihedral group runs in sub-exponential time $\widetilde{O}(3^{\sqrt{2\log_3 N}})$. Known improvements on these constructions are due to Regev [Reg04] and again Kuperberg [Kup13] where the total computation time is estimated to be $\widetilde{O}(2^{\sqrt{2\log_2 N}})$. Table 1 provides the group details (i.e., domain and operation) and specific application of HSP for different groups and well-known applications. For example, solving the HSP for the group of the integer numbers domain ($\mathbb{Z}_N$) with the addition ($+$) operation and (0) as the neutral element can be used in RSA factorization, while ECDSA and El-Algamal algorithms can be broken by solving the HSP for the group identified by the $\mathbb{Z}_N \times \mathbb{Z}_N$ domain with the addition ($+$) operation. In this case, the operation is the component-wise addition, and the neutral element is the pair (0,0).

Table 1 - List of Hidden Subgroup Problem definition and their applications

HSP Group	Operation	Application
$\{0,1\}^n$	XOR	Simon's Algorithm
$\mathbb{Z}_N$	$+ \ mod \ N$	Period Finding Function
$\mathbb{Z}_N$	$+$	Shor's Factoring Algorithm (RSA)
$\mathbb{Z}_N$ x $\mathbb{Z}_N$	$+$	Shor's Discrete Logs (ECDSA, El Algamal)
"Dihedral Group"	Composition of Symmetries (rotations, reflections)	Approximate SVP (and CVP)

2.2. Quantum-Resistant Cryptograpy

As we have seen, lattice-based cryptography does not come, so far, with efficient algorithms, quantum or classic, that can solve the underlying problem efficiently. That is why some of the most promising algorithms that are still present in the final round of the NIST competition are lattice-based. These mathematical objects are, in practice, regular collection of equally spaced vectors or points. In other words, lattices are regular arrays (or grids) of points that are generated by a combination of basis vectors. Lattice-based cryptography properties are rooted in the hardness of solving certain topological problems for which we do not have an efficient algorithm for, like finding the Shortest Vector Problem (SVP) or the Closest Vector Problem (CVP) given a specific basis for the lattice. Algorithms like Falcon [Fa17] or Dilithium-Crystals [Di17] fall in this category and produce the smallest authentication traces overall (i.e., signatures range from 700 bytes to 3300 bytes).

Another class of algorithms to keep an eye on is the Isogenies-based ones. These algorithms use a different structure than lattices and have been proposed for key-exchange algorithms, namely Key

Encapsulation Mechanisms or KEMs. Specifically, isogeny-based cryptography combines morphisms (or isogenies) among elliptic curves to provide Perfect Forward Secrecy (PFS) properties. Although Isogeny-based cryptography is computationally very heavy, it uses the shortest keys in the post-quantum algorithm landscape. SIKE is an example of such a class of algorithms.

Together with these two classes of algorithms, there is another type of algorithm that should be kept in our minds as a possible alternative: hash-based signature schemes. These algorithms rely on very different security property and data structures that are not tractable via HSP. The main issue with stateless hash-based schemes is the size of signatures: the lack of structure in the data comes at the expense of very large cryptographic signatures (although public keys are extremely small). Although the size of signatures hinders, today, their adoption, the security of this class of algorithms is not affected by advancements in HSP solving for non-commutative groups. A well-known hash-based algorithm that will probably be re-included in the NIST standardization process because of its security properties is SPHINCS+ [Sp15].

3. Multi-Keys Trust Environments

X.509 certificates have, so far, been used to link one public key to a single identity. This is true for Trust Anchors (or TAs), Intermediate CAs (or ICAs), and End-Entities (or EE). However, because the encoding of public key data structures inside certificates depends solely on the specific OID used to identify them, the inner BIT STRING that encodes the key value can be re-engineered to accommodate for any data structure. In our original work that was presented at SCTE Tech Expo 2020, we used the algorithm agility feature built in into X.509 certificates and defined a new OID to identify a key structure which implements a `SEQUENCE of SubjectPublicKeyInfo` structures. Each of the structures in the Composite Key encodes a specific public key which encompass the algorithm identifier together with parameters and the key value.

Practically, 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 }
```

The `compositeSignatures` identifier is used to identify the type of signature, and the corresponding value, encoded in the `signatureValue` field, contains multiple signatures and associated parameters. Each of these individual `SignatureInfo` entries carry the information about one of the signatures applied to the certificate in the same order the corresponding public keys appear in the multi-key issuer's certificate.

3.1. Current Limitations

When we first drafted the public release of Composite Crypto, there were still few unresolved issues that were associated with the use of multiple keys in a single certificate. The main issues were related to (a) handling error conditions when only some of the signatures are reported to be bad, (b) the complexity of enabling encryption in multi-key environments, and (c) how to handle ecosystem-wide algorithm revocation.

At the time of publication, specifically, some argued that, although using multiple keys of different type is a valuable feature (not only when it comes to backward compatibility or future-looking deployments), the complications introduced from the use of multiple keys to validate a single object required the deployment of some complex validation policy and additional infrastructure elements. At the same time, a second argument against the standardization of multi-key certificates was related to the impracticality of modifying current crypto libraries to accommodate for new types of error conditions and API changes. A final argument against our idea was based on the difficulty of guaranteeing the correct distribution of validation policies across entire ecosystems, like the broadband one, without the need for deploying additional infrastructure elements.

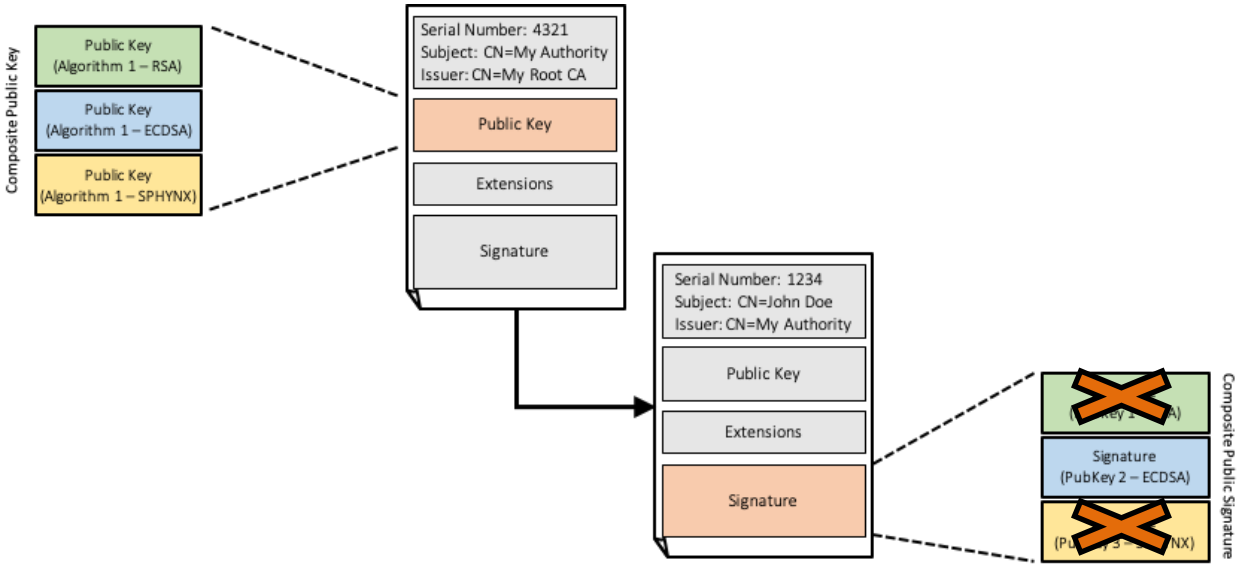

Figure 1 - Example of new error paths introduced with the use of multiple keys

Figure 1 sketches an explanatory error scenario where a composite signature, in this case on a certificate, has only one valid signature component. In our original work, the decision about the overall validity of the signature was left to the crypto library. This ambiguity posed a serious issue for consistency in how applications deal with these new mixed error states. In some scenarios, you want the possibility for the components of a composite signature to be "alternatives" so that a relying party can use the keys they prefer and/or understand (any of the signatures are equivalent). In other situations, instead, you want the possibility to report the signature to be valid only if all the components of signatures verify correctly. In other words, by providing an underspecified behavior, we inadvertently introduced, from an ecosystem perspective, the possibility for unpredictable results.

A problem that did not have a solution until now.

Another aspect that must be considered for signature validation is the level of trust in the algorithm throughout time. In the above example, let's imagine that two out of three components of the signature are reported to be erroneous. Let's also imaging that at time t_0, the use of ECDSA alone provides enough security for the identified application and ecosystem. Let's now move the clock 3 years forward at a time

$t_1 = t_0 + 3$ years. Is the signature still to be considered valid? Without additional indications from a trusted source, applications and users are faced with an impossible task that cannot be easily resolved.

The next section explains how we solved the identified problems by introducing a second data structure that explicitly defines the relationship across the different key and signature components.

4. Composite Crypto vs. Combined Crypto

As described in the previous section, the fact that there were no clear semantics associated with the Composite Crypto was the source of the issues we were facing. This was reflected by the need of requiring crypto libraries to change their current APIs to support different validation policies. Because this might not be practical and might hinder adoption, we defined an alternative mechanism to drive the deterministic behavior when validating multi-key signatures. The core of our solution turned out to be extremely simple: defining, on top of the current ones, a new set of algorithm identifiers for keys and signatures that we call *Combined Crypto*.

With the introduction of these new OIDs (one for `subjectPublicKeyInfo` identifiers and one for `Signature` identifiers), we now have the possibility to explicitly define, via OIDs, the logic operations that crypto libraries must apply when validating the multi-key signatures.

When the *Composite Crypto* identifiers (also referred to as `compositeOr`) are used, the relationship across the different signatures is a logical “OR”. This means that the crypto library can use ANY of the signature components to determine the validity of the entire signature. A simple deterministic algorithm can be defined that goes through the list of signatures and stops at the first correctly validated one. If no signatures are correctly validated (i.e., because the values are corrupted or because the entity does not support the specific algorithm), the overall signature is not valid.

When the *Combined Crypto* identifiers are used instead, the relationship across the different signatures is a logical “AND”. This means that the crypto library must, in this case, positively verify ALL the signature components before being able to declare the whole signature valid. Also in this case, a simple deterministic algorithm can go through the list of signatures and stop at the first incorrectly validated one. If even one signature is not correctly validated (e.g., because of erroneous calculations or unsupported algorithm), the overall signature is not valid.

4.1. Advanced Key Structures

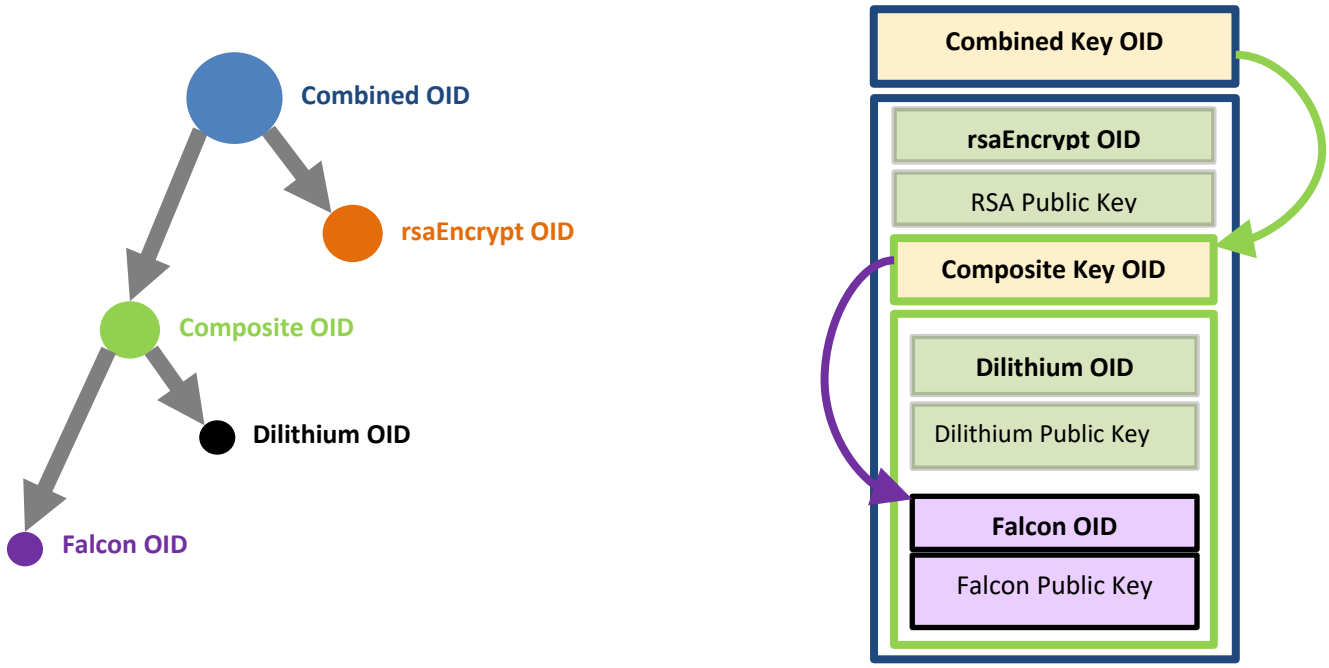

Figure 2 - Tree Representation of a Multi-Key Public Key Info structure

Figure 2 provides a tree representation of a key structure where the use of the "AND" and "OR" logical functions are leveraged. In the provided example, the represented key structure mandates for the use of an RSA component in the Combined Key container together with one of the two components from the Composite Key one. In fact, as previously discussed, when validating the components of a Combined key (as in this example), all of them must be validated correctly and that requires both the RSA and the Composite Key signatures to be valid. Back to the specific example, this translated to the need for the RSA signature to be valid together with, at least, one of the components in the Composite Key, i.e., the Dilithium or the Falcon signatures.

By following the described approach, CAs and PAs can design their certificate profiles with specific key structures for their certificates with deterministic behavior. For example, PKI architects can now decide to use a classic algorithm as the first element in a Composite Key to maximize backward compatibility. Another optimization strategy could be to maximize efficiency by using the fastest algorithm, from a validation standpoint, as the first element in composite keys, while using other FIPS and/or non-FIPS algorithms in combined keys to enforce the use of both types of cryptography together (i.e., classic and post-quantum).

4.2. A Deterministic Algorithm for Multi-Key Signature Validations

When using multi-key certificates, the `subjectPublicKeyInfo` structure of the certificate can have two different types of OIDs. The first type of OIDs is a container OID (i.e., the Composite or Combined ones) while the second type is a "real" algorithm OID such as, for example, rsaEncryption. Equation 1 provides a deterministic algorithm for validating multi-key signatures in pseudo programming language.

```
FUNCTION: Validate Signature Component
------------------------------------------------------------------------
If Signature OID is Combined:
    For Each Component in Combined:
        If Signature Component is Combined:
            ERROR: Recursion
        End If
        If Validate Signature Component is NOT Valid
            Return False
        End If
    End For
    Return True
Else
    If Signature OID is Composite:
        For Each Component in Composite:
            If Signature OID is Composite:
                ERROR: Recursion
            End If
            If Validate Signature Component is Valid
                Return True
            End If
        End For
        Return False
    Else
        If Validate Signature Component is Valid
            Return True
        Else
            Return False
        End If
    End If
```

Equation 1 - Multi-Key Signature Validation Algorithm

For each of the nested components in combined signatures we evaluate it. We stop the validation process as soon as one component does not verify correctly. In this case the full combined key is invalid. On the other hand, if all components of the combined signature verify correctly, the combined key is considered valid.

For each of the nested components in composite signatures, we also evaluate it. However, differently from the combined key container, in this case we consider the composite key valid if at least one of the components is valid. The algorithm goes through the list of components and considers the composite key valid as soon as one component validates correctly. Conversely, the composite key is invalid if all the components (and not just one as in the combined key case) do not validate correctly.

5. Algorithm Revocation Via CRLs and OCSP Responses

With the possibility of algorithms being completely compromised overnight by quantum computers, PKIs are faced with a new problem: distrusting certificates that use a specific public key algorithm. Independently from the use of multi-key or single-key certificates, the inability, today, to provide such a mass-revocation tool can hinder our ability to effectively revoke the use of an algorithm.

---

*We are missing, today, an important tool in PKIs that is relevant for Post-Quantum algorithms deployment efforts, and that is **Algorithm Revocation**.*

We can easily see the impact of the lack of such tool with the latest example of algorithm deprecation that required a long time to complete (SHA-1). Specifically, when looking at the evolution of the deprecation process, we notice how it has happened very slowly and its resolution used ad-hoc criteria and deployment strategies instead of delivering formal ways to revoke its use across entire ecosystems. As a result, Certification Authorities, although they are there to guide the ecosystem and have the authority to revoke identities as needed, they still lack practical tools and options to enforce algorithm deprecation.

5.1. Policy Authorities as Sources of Trust

Since we introduced the concept of algorithm revocation in conjunction with multi-key environments, some critiques have been directed at the chosen trust model arguing that an external authority should be the one to provide algorithm deprecation information. Because this is an important governance principle, we want to provide additional considerations that can help understanding the principles we rely on when extending existing revocation mechanisms.

The trust model that is usually assumed in PKIs mandates for CAs to keep all participants in the ecosystem behaving according to a common policy. Therefore, CAs are already entities with a clear mandate to protect the integrity of the ecosystem by following verifiable procedures - this includes the possibility to revoke certificates. CAs are, therefore, the entities that, in accordance to defined policies, should provide indications about which type of keys should not be trusted throughout the PKI lifecycle.

In a trust infrastructure, besides the set of CAs that provide their services to the community, it is common practice to deploy a Policy Authority that is responsible for the ecosystem Certificate Policy (CP). When available, the content of the policy document is used to align CAs requirements across the whole ecosystem. In the DOCSIS PKI, the Policy Authority is operated by CableLabs on behalf of the entire ecosystem and is appointed with the task of making sure that the whole PKI is secure. As this governance model has been successfully exported to other ecosystems of interest for the broadband industry, our work can be extended and adopted in other device-centric ecosystems where a common trust infrastructure enables interesting and effective crypto-migration strategies (e.g., Wi-Fi Alliance/Passport 2.0, CBRS-A, etc.)

5.2. Algorithm Revocation vs. Certificate Revocation

When considering revocation and its practical impact over the ecosystem, an important consideration to make is related to the scalability of algorithm revocation vs. certificate revocation. Today, when a crypto algorithm needs to be replaced because of possible security risks or compromises, CAs must revoke every single affected certificate to make sure that the faulty algorithm is not used anymore.

Even when the higher levels of the PKI are protected with quantum safe algorithms (i.e., Root and Intermediate CAs), the option of using traditional revocation mechanism, i.e., via serial numbers, comes with very high costs related to adding many certificates to the revoked lists – both CRLs and OCSP servers are negatively impacted by these massive revocation events and can easily collapse under this added load (e.g., in the DOCSIS PKI the number of active certificates to revoke can be in the hundreds of millions). Conversely, the revocation mechanism described in this invention provides a very efficient way to mass-revoke certificates when and if needed. The mechanism is lightweight both on the Certificate Service Providers (or CSPs) when creating and distributing this information, and on the client when validating certificate chains and signatures. The rest of this Section provides a detailed description of the data structures, procedures, and extensions we defined to enable algorithm revocation.

5.3. Algorithm Revocation and Multi-Key Environments

The lack of standardized secure mechanisms to provide algorithm revocation is not a new problem. However, with the introduction of multiple keys within a single certificate, the problem of algorithm revocation becomes more evident.

In our work we focus on revocation of key structures, rather than a simply focusing on algorithms, to provide the possibility to better manage algorithm trust. For example, there might be the need to revoke a specific key configuration across the whole set of issued certificates (i.e., a specific algorithm or a specific algorithm hierarchy) without having to completely revoke its use in other cases. To address all these use cases, our work allows the ecosystem administrators to revoke, for example, the use of RSA as a primary key type in certificates or within Composite Key containers, but still allow the use of the RSA algorithm when used as a component of a Combined Keys (e.g., RSA + Post-Quantum Algorithm).

An example of the complexity that raises with the introduction of multi-key certificates can be easily shown by looking at the evolution of algorithm deprecation over an extended period. Let's imagine the case where a relying party correctly verifies the signatures on a specific document or certificate and let's also imagine that this process is repeated over and over – while the validation results do not change, the trust in the algorithm itself can change. For example, one of the algorithms used in the PKI might be compromised or there might be a new requirement, during uncertainty, to prevent the use of classic and/or post-quantum algorithms by themselves (e.g., you must use a combined RSA and Falcon signature).

The mechanism described in this work addresses all these cases.

5.4. Key Configuration Revocation

The key configuration revocation mechanism we introduce in this paper focuses on revoking specific key configurations via the use of key configuration revocation lists. To this purpose, we needed to provide a way to identify key configurations that must be distrusted. As discussed in Section 4.1, we can represent

key configurations as binary trees where the Composite and Combined nodes provide the bifurcations in the tree structure, while the individual components represent the end nodes (or leaves) of the tree. Few example configurations are provided in Figure 4.

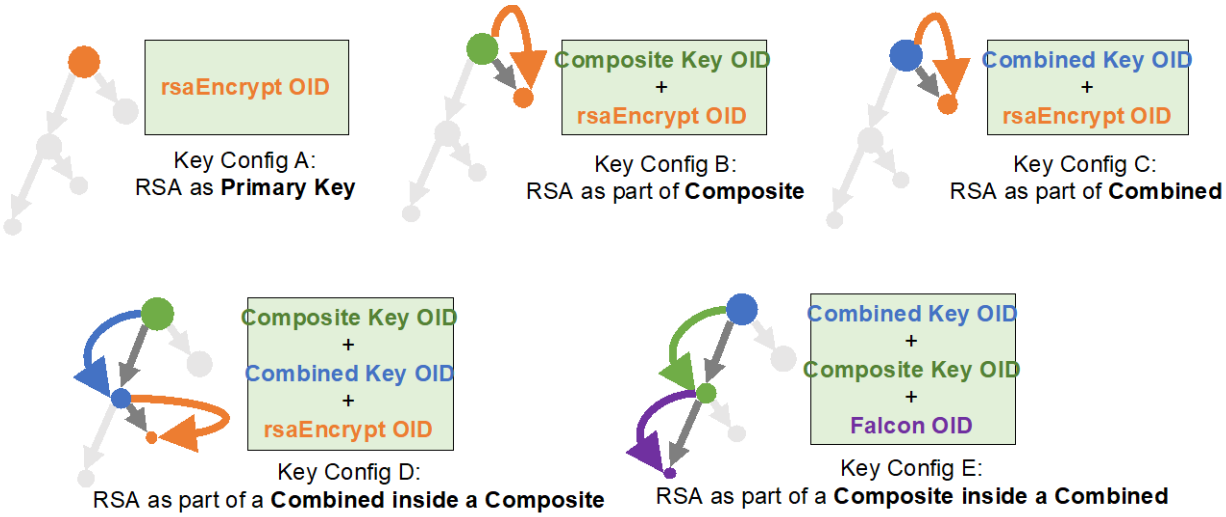

Figure 3 - Example Key Configurations for different types of primary algorithms.

For each of the key configurations that the CA wants to deprecate or revoke (or is instructed to do so by the PA), the CA generates a sequence of OIDs that we refer to as the `KeyConfigRevocationData`. This list of algorithm OIDs is then embedded as the value of an extension in OCSP responses and CRLs that are issued from the CA (or the delegated signer). The value of the `KeyConfigRevocationList` extension is implemented as a `SEQUENCE OF KeyConfigRevocationData`.

More specifically, each of these entries provides information about how to uniquely identify the specific key configuration (e.g., "(Start) → RSA" or "(Start) → CompositeCrypto → RSA"). Additionally, it is possible to specify an optional trust period for the algorithm in the form of `doNotUseBeforeDate` and `doNotTrustAferDate` fields.

The `KeyConfigRevocationList` data structure and associated identifier(s) are defined as follows:

```
keyConfigRevocationList-id OBJECT IDENTIFIER ::=
         {iso(1) identified-organization(3) dod(6) internet(1)
          private(4) enterprise(1) OpenCA(18227) 13 }

KeyConfigId ::= 1..MAX OF OBJECT_IDENTIFIER

KeyConfigRevocationData  ::=  SEQUENCE  {
        keyConfig               KeyConfigId,
            --- Identifier of the specific Key Configuration
            --- identified by this data structure
        doNotUseBeforeDate    [0]  GENERALIZED_TIME      OPTIONAL,
            --- Time before which the key configuration
            --- should not be used
        doNotTrustAferDate    [1] GENERALIZED_TIME     OPTIONAL,
            --- Timestamp after which the key configuration
            --- identified by keyConfig should not be trusted
            --- by the ecosystem clients anymore }

KeyConfigRevocationList  ::= SEQUENCE (1..MAX) OF KeyConfigRevocationData
```

To deprecate a specific algorithm when validating certificates (e.g., RSA), the data structure of the key revocation extension (i.e., the `keyConfigRevocationList`) carries the specific algorithm identifier as the only value in the `keyConfig` field. This configuration would not prevent, however, the use of the identified algorithm inside Composite or Combined keys because the algorithm identifier's list would be different. To deprecate both the use of an algorithm as a primary key in the certificate and as a component of Composite Keys (but leaving the possibility to leverage it in a Combined Key), the CA would generate two entries. The first one carries a sequence that comprises only a single identifier, e.g., the RSA algorithm identifier. This sequence deprecates the use of the algorithm as a primary key. The second one carries the sequence "Composite Crypto OID → RSA algorithm OID". This sequence deprecates the use of the algorithm as a component of Composite Keys (i.e., using RSA inside Composite Crypto keys).

5.5. Deprecating the use of multi-key certificates

CAs might also need a mechanism to deprecate the use of Composite Crypto or Combined Crypto within the ecosystem for when, for example, a transitioning period is over, and infrastructures and devices have fully transitioned to the new algorithms.

In this case, no additional mechanisms are required because the very same approach described in this paper can also be used to deprecate multi-key certificates: the CA generates a `KeyConfigRevocationData` entry where the `keyConfigId` carries only the Composite Crypto or the Combined Crypto object identifier(s) as needed.

6. Solving the Multi-Key Encryption Conundrum

Multi-key environments can provide interesting options to address encryption under today's cryptographic uncertainties. For this discussion, we choose the use case that deals with encrypting a document for a specific recipient as the explanatory relevant use-case. Specifically, the open problem we are focusing on is how to determine which key or set of keys should be used to encrypt a document for a recipient in the presence of multiple certificates and algorithms.

Similarly to the algorithm revocation case, linking multiple keys to the same identity is not a new problem and still we have no standardized solutions for it. In fact, there is no accepted procedure, today, to securely link together identities contained in different certificates that might even be issued from different CAs or different PKIs.

6.1. Encryption, Certificates, and Multiple Algorithm Support

To better explain the issues that crypto libraries and applications need to address when supporting multiple algorithms to encrypt data, let's go back to our example and describe the process of encrypting a document that is to be shared with a single recipient. In our example, let's assume that multiple algorithm support (e.g., RSA and Dilithium-Crystals) is required but only single-key certificates are deployed. This can happen, for example, when encrypting an e-mail for a recipient that might have multiple certificates, i.e., one with an RSA key and another with a Dilithium-Crystals one. For brevity and clarity, in the rest of the discussion we omit the description of the procedures for encrypting the data via a symmetric algorithm (not relevant for our discussion) and focus on the differences, when considering the encryption process, between single-key and multi-key certificates.

Before encrypting, applications must validate the revocation status of the recipient's certificate by accessing the certificates' revocation information (i.e., CRLs or OCSP responses) from the appropriate URL for all the certificates in the validation chain of the recipient. This is an essential step that prevents the leakage of the encrypted information for compromised certificates or keys. Without any indication of what the status of the algorithm (or key configuration) is or might be in the future, applications will happily encrypt the data for each of the certificates and possibly leak the encrypted content if one of the algorithms is broken.

This simple example shows the two main issues that the industry faces under the current crypto uncertainty when single-key certificates are used: dealing with the inefficiency of using multiple certificates connected to a single identity (i.e., need to interact with multiple infrastructures/services for a single encryption/validation operation) and the inability of efficiently communicating how to leverage the security of multiple algorithms together (i.e., "AND" or "OR" operations).

When looking at the first issue, multi-key certificates provide a distinct advantage: the need for less queries to the infrastructure. Specifically, because applications have to validate only one certificate chain per recipient, the number of requests to OCSP and CRL servers is greatly reduced. For example, in a three-tier infrastructure (i.e., Root CA, Intermediate CA, End-Entities) with three different algorithms deployed via single-key certificates, applications might need to perform up to six different OCSP or CRL queries and securely store 3 different Root CAs, while when multi-certificates are used, applications might need up to only 2 different queries and securely store a single Root CA. When looking at the second issue, the application that is performing the encryption is faced with the same uncertainty we noticed in the first formulation of our composite cryptography proposal (i.e., lack of deterministic behavior) because there is no possibility to dictate if the keys in the different certificates are equivalent or if they must be used together.

Ultimately, this one-to-one paradigm (i.e., one key for one identity) is also reflected everywhere in X.509 trust infrastructures where the assumption is that different certificates are associated with possibly different identities. Multi-key certificates solve the underlying conundrum by using a single identity, thus enabling the use of multiple algorithms across the board: from network functions to document signing.

6.2. More Efficient Encryption Process with Multi-Key Certificates

As described earlier, the ambiguity that was introduced with the initial proposal for multi-key certificates is completely resolved in this work by using explicit logic operations across keys and signatures that are completely defined by the specific OID used (Composite or Combined). Also in the encryption case, we leverage the separation of "OR" and "AND" logic operations to provide crypto libraries with deterministic encryption and decryption behavior. Table 2 provides a summary of the differences between Composite and Combined crypto when it comes to encryption options. Specifically, a Composite Key is enabled for encryption if at least one of the components algorithms supports encryption while a Combined Key is enabled for encryption if all the components' algorithms support encryption.

Back to our example, by providing algorithm deprecation information together with certificate revocation information, the encryption process can be performed even more securely than we do today and increase flexibility by supporting forward-looking or backward-compatible key structures. Even

outside the multi-certificate use-case, the availability and use of key configuration deprecation enhances the security of the whole ecosystem and help to prevent possible data breaches.

Table 2 - Encryption Operations for Composite and Combined Crypto

Composite Crypto	Combined Crypto
When Encrypting for a Composite Key, the encryption is performed with ***all the public keys SEPARATELY***	When Encrypting for a Combined Key, the encryption is performed with all the ***keys in a COMBINED way***
When Decrypting with a Composite Key, the decryption ***can be performed with ANY of the private keys*** related to the single public key components (OR)	When Decrypting with a Combined Key, the ***decryption must be performed with ALL the private keys*** related to the single Public Key components (AND)

7. Conclusions and Future Work

In this work we provide a description of the latest results when it comes to Composite Crypto and deployment of post-quantum algorithms. Specifically, we extend our original proposal to address the origin of the processing uncertainty that affected our original proposal: an incomplete design.

By adding a new set of OIDs, we can now express what the relationship across signatures (or keys) should be, thus providing a deterministic validation and encryption process. This simple enhancement unlocks deterministic behavior for crypto libraries without the need for deploying complex validation policies as it was initially envisioned. In other words, with the discussed new additions to our framework, the key structure of multi-key certificates itself provides clear validation, encryption, and decryption processing rules for crypto libraries.

On top of these important results, we identified CRLs and OCSP responses as the preferred mechanism to carry sequences of OIDs (and validity periods) to deprecate individual key configurations. This mechanism for algorithm revocation can be used in conjunction with both single key and multi key certificate environments.

Ultimately, the considerations contained throughout the paper show that the use of multi-key certificates can lower the cost of multiple algorithm deployment and provide the possibility to better manage, at the ecosystem level, the risks related to cryptographic failures. As we continue to evolve tools and specifications for multi-key environments, we envision that their deployment might become a common mechanism for delivering dynamic crypto-agile ecosystems in the future and, at the same time, simplifying new algorithm deployments and support algorithm migrations processes.

Abbreviations

CA	certification authority
CBRS-A	citizens broadband radio service alliance
CRL	certificate revocation list

CSP	certificate service provider
CVP	closest vector problem
DER	Distinguished Encoding Rules
DOCSIS	Data Over Cable Service Interface Specifications
DH	Diffie-Hellman
EC	Elliptic-Curves
ECDSA	Elliptic-Curves Digital Signing Algorithm
EE	end entity
FIPS	Federal information processing standard
HSP	hidden subgroup problem
ICA	intermediate certification authority
I-D	internet draft
IETF	Internet Engineering Task Force Standards Organization
KEM	key encapsulation mechanism
KEX	key exchange (algorithm)
NIST	National Institute of Standards and Technologies
PA	Policy Authority
PKC	public-key cryptography
PKI	public-key infrastructure
OCSP	online certificate status protocol
OID	object identifier
PFS	perfect forward secrecy
PQ	Post quantum
PQA	post-quantum algorithm
QC	quantum computing
R-PHY	Remote RF Layer (PHY)
RSA	Rivest-Shamir-Adleman (cryptosystem)
SHA-1	Secure Hash Algorithm (160 bits)
SCTE	Society of Cable Telecommunications Engineers
SVP	Shortest vector problem
TA	trust anchor
TLS	Transport Layer Security
SCTE	Society of Cable Telecommunications Engineers
S/MIME	secure e-mail message format
Wi-Fi	wireless
X.509	standard format for digital certificates
XOR	exclusive OR operator

Bibliography & References

[Ec05] American National Standards Institute, *Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI X9.62, November 2005.

[Rsa16] The Internet Engineering Task Force (IETF) – IETF RFC 8017. PKCS #1: RSA Cryptography Specifications Version 2.2, edited by K. Moriarty et al., November 2016. Also available at https://datatracker.ietf.org/doc/rfc8017/

[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.

[X509] 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.

[Pala04] The Internet Engineering Task Force (IETF) – I-D draft-ounsworth-pq-composite-sigs-04 - Composite Keys and Signatures For Use In Internet PKI, edited by M. Ounsworth and M.Pala, Jan 2021. Also available at https://datatracker.ietf.org/doc/draft-ounsworth-pq-composite-sigs/

[Shor97] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: SIAM Journal on Computing 26.5 (Okt. 1997), S. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137 / S0097539795293172. arXiv: quant-ph/9508027.

[Reg02] Regev, O. “Quantum computation and lattice problems.” The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings. (2002): 520-529.

[HeHø04] Mark Ettinger, Peter Høyer, Emanuel Knill, The quantum query complexity of the hidden subgroup problem is polynomial, Information Processing Letters 91 (1) (2004) 43–48.

[Reg04] *Regev, O.. “A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space.” arXiv: Quantum Physics (2004).*

[Kup13] Kuperberg, G.. “Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem.” TQC (2013).

[Fa17] Falcon - Fast Fourier Lattice-based Compact Signatures over NTRU. https://falcon-sign.info.

[Di17] Dilithium-Crystals - Dilithium digital signature scheme. https://pq-crystals.org/dilithium/.

[Sp15] SPHINCS+ Stateless hash-based signature algorithm website. https://sphincs.org.

