A Secret Key Spreading Protocol for Extending ETSI Quantum Key Distribution*

Thomas Prévost[1][a], Bruno Martin[1][b] and Olivier Alibart[2][c]
[1]*I3S, Université Côte d'Azur, CNRS, Sophia-Antipolis, France*
[2]*InPhyNi, Université Côte d'Azur, CNRS, Nice, France*
{*thomas.prevost, bruno.martin*}*@univ-cotedazur.fr, olivier.alibart@univ-cotedazur.fr*

Keywords: ETSI, Quantum Cryptography and Safety, Formal Verification, ProVerif, Maude, Shamir's Secret Sharing, Network Security.

Abstract: This paper presents an information theoretic secure secret key transfer protocol by Quantum Key Distribution (QKD) in the case of multi-hops quantum links between the two correspondents. We aim to transmit a secret between two parties using existing quantum infrastructure in the case where half of the intermediate routers are evil. We recursively divide the secret into Shamir's shares, which are transmitted through different routes. This protocol has been successfully verified with ProVerif, which grants the secrecy of the transmitted key. We also propose an on-the-fly route discovery algorithm, in case the network is too large for each node to know all possible routes, and provide a formal verification of this algorithm using Maude.

1 INTRODUCTION

Key exchange protocols enable two previously unacquainted parties to establish a common cryptographic key for symmetric encryption. Traditional methods rely on one-way cryptographic functions, but current asymmetric algorithms (e.g., RSA, ECC) face obsolescence with the advent of quantum computers (Bhatia and Ramkumar, 2020). Post-quantum algorithms promise resistance to quantum attacks but remain unproven, with vulnerabilities still possible (Kaluderovic, 2022). Moreover, computational security offers only time-limited guarantees.

Quantum Key Distribution (QKD) leverages quantum physics, specifically the non-cloning theorem (Zygelman and Zygelman, 2018), to detect eavesdropping via alterations in qubit states (e.g., photon polarization). Unlike computational methods, QKD ensures security through real-time attack detection. However, QKD requires a pre-established authenticated classical communication channel, often secured via public-key cryptography, to guarantee forward secrecy. In QKD systems, photons are transmitted through "black fiber" quantum links, supported by authenticated classical links for data exchange.

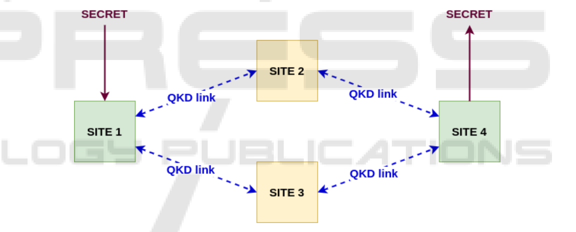

Figure 1: Given an infrastructure with existing QKD links between sites, we aim to transmit secret between the green sites that are not directly connected via a QKD link.

QKD networks face distance limitations, necessitating multi-hop connections (Fig.1) for secure key exchange across data centers, a scenario relevant to Cloud providers. (ETSI, 2019) proposed a QKD key transmission standard but left technical aspects, such as multi-link key exchange, unaddressed.

Existing multi-path QKD protocols (Wang et al., 2023; Vyas and Mendes, 2024; Mehic et al., 2020; Salvail et al., 2010) rely on XOR operations for information-theoretic secrecy. (Choi et al., 2021) reviews QKD communication protocol standardization.

We propose a novel protocol using recursive Shamir's secret sharing to securely extend QKD-based secret transmission across broader networks.

[a] https://orcid.org/0009-0000-2224-8574
[b] https://orcid.org/0000-0002-0048-5197
[c] https://orcid.org/0000-0003-4404-4067
* This work was supported by a government grant managed by the Agence Nationale de la Recherche under the Investissement d'avenir program, ref. ANR-17-EURE-004.

Prévost, T., Martin, B. and Alibart, O.
A Secret Key Spreading Protocol for Extending ETSI Quantum Key Distribution.
DOI: 10.5220/0013077100003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 411-418
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

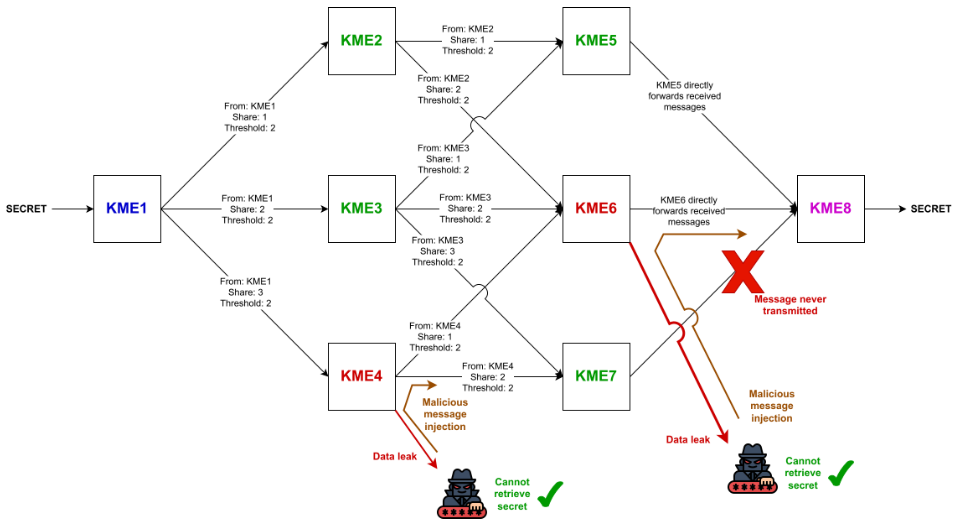

Figure 2: Example of a QKD network modeled and proved with ProVerif with n = 8 participants and a threshold of k = 2. In this example KMEs 4 and 6 are controlled by the attacker, and the link between KME 7 and KME 8 has been cut.

This approach ensures perfect secrecy and robustness against node corruption or link failures. It supports dynamic route discovery in large networks via a protocol verified in Maude (Clavel et al., 2003). Additionally, ProVerif-based formal security proofs validate our method.

This paper is organized as follows: Section 2 reviews the ETSI GS QKD 014 standard. Section 3 details our secret transfer protocol. In section 4, we provide a formal verification of our protocol using ProVerif. Section 5 presents our dynamic routing protocol and security analysis.

2 THE ETSI GS QKD 014 STANDARD PROPOSAL

In the (ETSI, 2019) protocol, two types of entities coexist: *Key Management Entities* (KME) in charge of quantum key exchange and storage, and *Secure Application Entities* (SAE). The latter correspond to final applications which will subsequently use the exchanged keys. The protocol considers that several SAEs can share the same KME within a "secure" zone, i.e. connected in a secure perimeter with only trusted nodes. A "secure" zone refers, for example, to a data center, in which the network is assumed to be secure. Within a secure zone, communications are carried out via a classical TLS secured link.

In order for the SAEs to be able to request a common key from their respective KMEs, it is necessary for the latter to carry out a secret exchange through a quantum link, via a QKD protocol. However, since it is a "quantum network", there is not necessarily a direct quantum link between two KMEs. The ETSI GS QKD 014 standard proposal does not detail how to transmit keys in this case: "*how KMEs relay keys securely in a QKD network is outside the scope of the present document*". Additionally, every KME is considered "trusted", which is a pretty strong assumption, but plausible in a context of cross-datacenter use : "*It is assumed that every Trusted Node is securely operated and managed.*". (Prévost et al., 2024) proved that under these assumptions, there is no security vulnerability in the standard protocol proposal by using ProVerif.

3 OUR NETWORK KEY TRANSMISSION ALGORITHM

3.1 Secret Sharing Scheme

The principle of Shared Secret Scheme consists of distributing a share of a given secret S_0 to each of n participants. To recover the secret, a subset of at least k participants must pool their shares, the threshold k having been defined in advance by the dealer. There are several secret sharing schemes, the best known being (Shamir, 1979) or (Blakley, 1979).

For Shamir's secret-sharing scheme, let n be the number of participants and k the threshold necessary to recover the secret S_0. Let $\mathbb{F}_p$ be the field with p elements, p prime, $p > S_0$. The dealer starts by generating a prime p randomly and a random polynomial of degree $k-1$, $P \in \mathbb{F}_p[X]$, $P(X) = S_0 + \sum_{i=1}^{k-1} a_i X^i$, where $a_1, \ldots, a_{k-1} \in \mathbb{F}_p$. The dealer generates n shares by distributing $S_1 = P(1)$, $S_2 = P(2)$, ..., $S_n =$

$P(n)$ to the n participants. A subset of k participants among n can retrieve the secret $S_0 = P(0)$ by reconstructing $P(X)$ with Lagrangian interpolation of the shares.

3.2 Algorithm Definition

We want to get rid of certain assumptions formulated in the ETSI standard proposal. We thus allow the possibility that certain KMEs could be corrupted by the attacker. This could happen if a malicious actor manages to break into a data center network. Our algorithm also supports the case when some quantum links could be broken. This is a scenario that can occur when an adversary physically cuts fiber links or launches a *Denial Of Service* (DOS) attack.

The algorithm relies on Shamir's secret sharing scheme, proven information-theoretic secure (Corniaux and Ghodosi, 2014), and therefore quantum safe.

Here, we focus on the case of a static network topology, which every KME would know in advance. If we want to introduce the possibility of dynamic routing, we would then have to ensure that an attacker cannot modify the knowledge that the KMEs have of the routes. We will address this issue in Section 5.

We introduce the concept of routing *layer*. For each secret we aim to transmit between an initial and a final KME, we can imagine the KMEs network as a *Directed Acyclic Graph* (DAG). A layer is thus a subset of nodes that will reshare the received shares with the same depth. Subsets of layers therefore differ depending on each secret transmission, since the routes will be different. In Fig. 2, the successive layers are:

Layer 1 = {KME 1}, Layer 2 = {KME 2, KME 3, KME 4}, Layer 3 = {KME 5, KME 6, KME 7}, Layer 4 = {KME 8}.

Here every possible path will pass once through one of the nodes of each layer. For example a possible route between KME 1 and KME 8 would be as follows: $KME\ 1 \rightarrow KME\ 2 \rightarrow KME\ 6 \rightarrow KME\ 8$.

Our algorithm also works with paths of different length. In Fig. 2 we represent a simplified case where each path has the same length.

For each KME the transmission of Shamir's shares is described in Algorithm 1.

So each node will generate new Shamir's shares from the received one and transmit them to the outgoing neighbor nodes. After receiving all the shares, the final KME recursively decrypts them, finally obtaining the secret distributed by the first KME.

We choose a threshold of 51% for each Shamir's sharing iteration. This means that an attacker would need to be able to control more than half of the KMEs in a layer of the network to be able to decrypt the secret, a very pessimistic estimate (as long as the network is well designed to avoid bottlenecks). With a lower threshold, the transfer of secrets would be faster and more resilient to network failures. Threshold can be set independently by each KME depending on the trust it places in those of the following layer.

Algorithm 1: Algorithm for transmitting secret shares, for each KME.

```
if is first KME then
    secret ← generate a secret;
    n ← recipient KMEs count // Number of outgoing neighbors;
    $k \leftarrow \llcorner n/2 \lrcorner + 1$ // Denotes the Shamir's threshold;
    shares ← Shamir's shares(secret, participants=$n$, threshold=$k$) // A list;
    for i in recipient_KMEs do
        Establish an encrypted communication with recipient_KMEs[i] using QKD;
        send shares[i] to recipient_KMEs[i];
    end
else
    forall share in received shares via QKD secured channel do
        if Only 1 KME left then
            Forward share to recipient_KME;
        else
            n ← recipient KMEs count // Number of outgoing neighbors;
            $k \leftarrow \llcorner n/2 \lrcorner + 1$ // Denotes the Shamir's threshold;
            shares ← Shamir's shares(share, participants=$n$, threshold=$k$) // A list;
            for i in recipient_KMEs do
                Establish an encrypted communication with recipient_KMEs[i] using QKD;
                send shares[i] to recipient_KMEs[i];
            end
        end
    end
end
```

3.3 Proof of Concept

A POC in `Rust` is available at: https://github.com/thomasarmel/qkd_kme_key_spread. The code runs entirely locally with logs explaining each step; it uses the ssss library for computing Shamir's secrets.

We have thus modeled the network topology presented in Fig. 2: KMEs 4 and 6 are controlled by the attacker and the link between KME 7 and 8 is broken. Note that the attacker always controls less than 51% of each routing layer.

3.4 Exchanged Data Volume

One might be concerned about the growth in messages' size as recursive sharing occurs in successive layers. In our implementations, every new iteration increases the shares size by 5 bytes. So for each KME, let l be the received share size and n the number of outgoing neighboring KMEs to which it will propagate the recursive shares, $L = n(l+5)$ is the total volume of data sent by the KME to the next layer:

Let C_1 be a cut between layers 1 and 2, C_2 between layers 2 and 3 and C_3 the last cut between layers 3 and 4. Consequently, in the network topology presented in Fig. 2, the total volume $V = V_{C_1} + V_{C_2} + V_{C_3} = 489$ bytes of messages exchanged for a key of initial size *l = 32 bytes*: $V = 3(l+5) + 7(l+5+5) + 2(l+5+5)$ *bytes*.

Taking into account that the connection between KME 7 and KME 8 is broken with no data exchanged.

As these messages are exchanged via classical links, this should not cause performance issues.

4 FORMAL VERIFICATION

4.1 ProVerif

The reader will be able to find in the project's GitHub repository a "`formal_verif`" folder, containing the formal proof of confidentiality and authenticity of the protocol. We used the ProVerif software, which translates the protocol into logical constraints and attempts to find a counterexample to prove the existence of an attack (Blanchet et al., 2018; Blanchet, 2012). The verification carried out by ProVerif is also proven to be complete, which means that there cannot be an attack that would not be discovered by the software.

ProVerif takes as input a formal description of a cryptographic protocol (Blanchet et al., 2016). Cryptographic primitives are represented by equations:

```
type key.
fun senc(bitstring, key): bitstring.
reduc forall m: bitstring, k: key;
  sdec(senc(m, k), k) = m.
```

This equation formalizes the fact that it is possible to call the decryption `sdec()` function with the result of the `senc()` encryption of a message m using the key k, and the same key k as argument. In this case only the message m will be retrieved. The cryptographic primitives are assumed to be perfect.

The attacker is represented as in the (Dolev and Yao, 1983) model, in which the verifier assumes that the attacker "carries the message". It is possible to specify whether the attacker can modify messages or not (resp. active or passive attacks).

User can formulate different queries to ProVerif, meaning different questions which the tool must answer, given the description of the protocol. The most common query is `query attacker(secret)` which should be understood as `query NOT attacker(secret)`, meaning ProVerif is asked to ensure that an attacker, as predefined before, is **NOT** able to recover `secret` in any way whatsoever.

Queries for material implication can be formulated, such as:

```
query var:type; event(second_event(var)) ==>
  event(first_event(var)).
```

This query checks that the `second_event` called with `var` variable, in all cases, implies `first_event`, called with the same variable. This type of query is very often used to verify the authentication property, since we ensure, for example, that the reception by a participant of a message necessarily implies its transmission by another participant.

Generally, the problem of proving a security protocol is undecidable. ProVerif can return the next responses for the queries:

- **True:** the query will succeed in all cases, which means for example that the attacker will not be able to retrieve the secret.
- **False:** the query failed and an attacker can retrieve the secret or create a counterexample to another query. In this case the tool returns the attack.
- **Cannot be Proven:** the tool cannot give a definitive answer. This happens when the protocol has not been modeled correctly or the amount of calculation to be carried out is too large.

Using ProVerif, we therefore attempted to prove confidentiality and authenticity of the network topology from Fig. 2. The result can be transposed to any other topology provided that the threshold of compromised nodes on each layer is not exceeded.

4.2 Assumptions and Modeling

ProVerif allows to model an active or passive attacker. As we do not wish to make dangerous assumptions, we have chosen an active attacker, capable of reading and modifying any message on the public channels.

We are not trying to prove the security of QKD here. We therefore consider that two KMEs which are directly connected by a quantum link have already established an encrypted and authenticated channel via QKD. We therefore model the communication between two successive KMEs by a private channel, i.e.:

```
free kme1_2_channel:channel [private].
```

To model a node controlled by the attacker, we consider that any information received will be brought to the attention of the latter. We have thus introduced a public channel through which the compromised KMEs leak all the received information:

```
free public_leak_channel:channel.
```

Furthermore, we consider that the attacker is able to compromise the messages sent from the compromised KMEs. The message flows output from these KMEs are therefore modeled by public channels, i.e. readable and modifiable by an attacker:

```
free kme4_6_dangerous_channel:channel.
```

Message authenticity is modeled by the following query: let *m* be a clear message, does the event of reception of *m* by KME 8 imply the unique transmission of *m* by KME 1?

```
event sent_secret_kme1(bitstring).
event received_secret_kme8(bitstring).
query s:bitstring;
  inj-event(received_secret_kme8(s)) ==>
    inj-event(sent_secret_kme1(s)).
```

The `inj-event` query checks the injective implication. With this query, we ensure that the event `received_secret_kme8` involves one and only one event `sent_secret_kme1`.

We chose to model only one iteration of the protocol, since we are modeling the transmission of a single secret across the network.

4.3 Analysis Results

The analysis of our protocol by ProVerif in the topology from Fig. 2 showed that it is safe, both for confidentiality and authenticity.

5 DISCOVERING THE ROUTES ON THE FLY

The main problem with the algorithm mentioned above is the need for each node to know exhaustively all the routes to all the other nodes. Since the network is more secure when there are many possible routes between two nodes, the total number of routes quickly becomes huge as the network grows. From a certain number of nodes and links in the network, the memory would be insufficient to store all the possible routes. It is therefore necessary to find a way for a node to exhaustively discover all the possible routes to another node on the fly when it wants to transmit a secret to the latter.

5.1 Attacker Model

An attacker who has managed to take control of an insufficient number of nodes to decrypt a secret would be interested in lowering the decryption threshold for the secret. Since this threshold is adjusted according to the number of possible routes (for example $> 51\%$ of possible routes), the attacker will have to mislead the attacked node about the possible routes to the destination node.

Here are the possible attacks we identified:

Route Suppression Attack. Make the attacked node believe that there are fewer possible routes than in reality, when the attacker took control of one of the nodes on a legitimate route.

Route Addition Attack. The attacker wrongly makes the attacked node believe that one of the nodes it controls is a possible route to the destination node.

Route Addition Attack with a Fake Link. The attacker has also taken control of another node that is really linked to the destination node. As in the **Route addition attack**, the attacker pretends a node it controls is a route to the destination, but the attacker uses a fake quantum link between the two nodes to forward the message to the destination. As the attacked node cannot know that the link between the two nodes controlled by the attacker is not quantum, the detection of this attack is very hard.

These maneuvers aim to give the nodes controlled by the attacker a Shamir share number higher or equal than the decryption threshold.

It is assumed that nodes can authenticate each other, even if they are not connected by a direct quantum link. PKC is used for this. Indeed, although public key encryption may be broken in the future, this is not currently the case. It is therefore possible to use public key authentication securely with PQC signature standards.

5.2 Description of the Exhaustive Route Discovery Algorithm

Technically speaking, our route discovery algorithm is more a flooding algorithm rather then an exhaustive path search algorithm. In fact, the starting node searches, among its neighbor nodes, which ones are a possible route to the destination without loops. While network route discovery has been widely studied, here we need to exhaustively discover all routes in order to ensure protocol security, so we will not be able to rely on those algorithms that search for the most efficient route.

We recall that the direct links between the nodes are already encrypted and authenticated by QKD. In order to authenticate the communications between the remote nodes, we will use a Public Key Infrastructure (PKI). Each node, or KME, having a unique identifier on the network, we can issue them a public key certificate, signed by a Certification Authority (CA).

When an initiator node wants to discover the possible paths to a destination, it asks all its direct neighbors if they are able to reach the destination node. These neighbors will then forward the question to their own neighbors etc. During each request, the list of previous nodes is transmitted to detect loops. The process stops when a loop is detected or the destination node is reached. At the end, each node is able to determine which of its neighbors is a possible route to the destination from the initiator. Since the direct links between nodes are authenticated, it is impossible for a node to claim to be the only possible route to the destination if this is not the case (**route suppression attack**).

In order to avoid the fictitious **route addition attack**, we add a direct link validation mechanism, as shown in Fig. 3. With its private key, the destination node certifies with a temporary certificate that there is indeed a direct path between it and each of its direct neighbors to the initiator, and then sends each neighbor their respective certificate. These same neighbors then generate a direct link certificate with their own neighbors to the initiator node, and send the certificate to their respective neighbors, in addition to the certificate received from the destination node. If a node receives several certificate suites to the destination node, it only keeps one, since it only has to prove that it is a possible route to the destination node. In the end, each node is thus able to prove that it is indeed a valid route between the initiator and the destination. The issuance of direct link certificates is not conditional on the communication being up at the time of the request. Thus, it is not possible for an adversary to cut a quantum link between two nodes in the hope of lowering Shamir's decryption threshold.

From a purely theoretical point of view, it is impossible to prevent an attack by adding a fictitious route between two malicious nodes (**route addition attack with a fake link**). Indeed, the initiator node has no way of verifying whether there is indeed a quantum link between two distant nodes. However, since dark fiber QKD has a maximum geographical distance, one could imagine implementing a heuristic aimed, for example, at detecting abnormally short paths. It would also be possible to detect if the adversary is trying to add virtual nodes that do not exist in order to add hops, since these nodes would not be connected to the rest of the network.

5.3 Formal Security Analysis

To ensure the security of our dynamic route discovery algorithm, we performed a formal security analysis. We still assume that cryptographic primitives are secure. In particular, we model authentication on direct links and between remote nodes as perfect. This is in practice a reasonable assumption; it is very unlikely that an attacker can break public key authentication while the protocol is running. In fact, we are adopting a "perfect forward secrecy" approach, meaning that the authentication mechanism is supposed to be constantly adapted to cope with the attacker's power, while encryption must be able to withstand changes in the attacker's power over the long term.

We used Maude (Clavel et al., 2003) to model our routing system. This is a reflective language and system supporting both equational and rewriting logic specification capable of automatically rewriting logical equations. We tested several initial network topology configurations that we modeled by equations, and we used Maude to simulate the different phases of the dynamic route discovery protocol. We thus tested that the final stable state corresponding to the configuration after the discovery ends corresponded to the expected state. We also simulated different initial configurations corresponding to possible attacks, to ensure that the final state did not correspond to a successful attack.

Here is the syntax in Maude to model a two nodes topology: $n(1, v(2, r(2)))$; $n(2, v(1, r(1)))$ where:

- $n()$ is a node. The global state is basically a set of nodes, separated by " ; ". Its parameters are: the node number $\in \mathbb{N}$ and the neighbors set.
- $v()$ is a neighbor. A node should contain a set of neighbors, separated by " : " with parameters a neighbor node number $\in \mathbb{N}$ and the set of possible routes (meaning reachable nodes) from this neighbor.

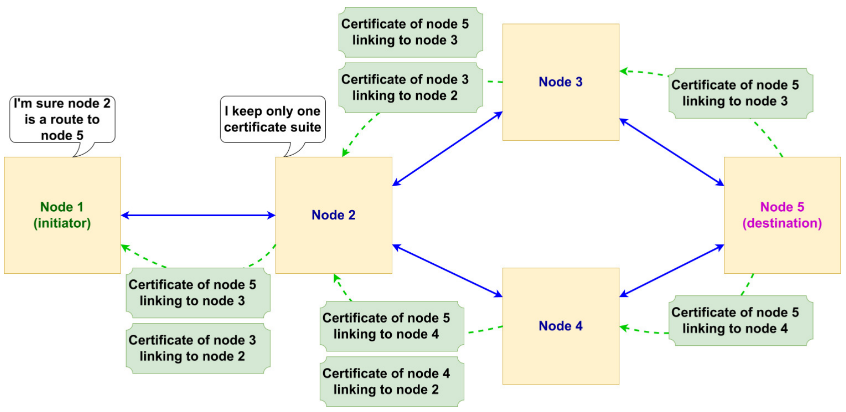
Certificate of node 5 linking to node 3
Certificate of node 3 linking to node 2
Node 3
Certificate of node 5 linking to node 3
I'm sure node 2 is a route to node 5
I keep only one certificate suite
Node 1 (initiator)
Node 2
Node 5 (destination)
Certificate of node 5 linking to node 3
Certificate of node 3 linking to node 2
Certificate of node 5 linking to node 4
Certificate of node 4 linking to node 2
Node 4
Certificate of node 5 linking to node 4

Figure 3: Example of forwarding the direct link certificate suite from the destination node to the initiator node. Since each node only needs to be able to prove that it is a valid route to the destination node, node 2 only needs to forward one certificate suite to node 1.

$$\begin{aligned} &\delta(\Omega \cup \{\, n(Y,\tau' \cup \{\, v(T,\chi' \cup \{\, r(Z)\,\})\,\})\,\} \cup \{\, n(X,\tau \cup \{\, v(Y,\chi)\,\})\,\}) && [1] \\ &= \begin{cases} \Omega \cup \{\, n(Y,\tau' \cup \{\, v(T,\chi' \cup \{\, r(Z)\,\})\,\})\,\} \cup \{\, n(X,\tau \cup \{\, v(Y,\chi \cup \{\, r(Z)\,\})\,\})\,\}, & [2] \\ \text{if } r(z) \notin \chi \text{ and } X \neq Z \text{ and } X \neq T & [3] \\ \Omega \cup \{\, n(Y,\tau' \cup \{\, v(T,\chi' \cup \{\, r(Z)\,\})\,\})\,\} \cup \{\, n(X,\tau \cup \{\, v(Y,\chi)\,\})\,\}, & [4] \\ \text{otherwise (no change)} & [5] \end{cases} \end{aligned} \tag{1}$$

1: Equation describing the evolution of the network state as modeled in Maude at the time of on-the-fly route discovery. Line [1] represents the δ function applied to the previous state of the network. Lines [2] and [3] show the net state of the network. Line [2] shows that node X associates with its neighbor Y the new route to Z if the condition in line [3] is met. Otherwise, the state of the network does not change, as shown in line [4].

- $r()$ used to designate a possible route from a neighbor. A neighbor should contain the exhaustive set of nodes it is able to reach. Possible routes are separated by "+". For a route to be validated, it is assumed that the node has performed the verification of the certificate suite. It has as parameter the route, meaning reachable node number $\in \mathbb{N}$.

This means that there are two nodes, node 1 and node 2. Node 1 has node 2 as a neighbor, and that neighbor is a valid route to node 2 (itself). Node 2 has node 1 as a neighbor, which is also a valid route to itself.

The route discovering algorithm equation as modeled in Maude is given in eq. 1, with:

- Ω a set of nodes (possibly empty)
- τ, τ' sets of neighbors (possibly empty)
- χ, χ' sets (possibly empty) of possible routes
- $T, X, Y, Z \in \mathbb{N}$ node numbers
- N set of nodes
- V set of neighbors
- R set of routes from a neighbor
- $n : \mathbb{N} \times \mathcal{P}(V) \to N$ the function describing a node
- $v : \mathbb{N} \times \mathcal{P}(R) \to V$ the function describing a node neighbor, as explained above
- $r : \mathbb{N} \to R$ the function describing a route from a neighbor, as explained above
- $\delta : \mathcal{P}(N) \to \mathcal{P}(N)$ the transition function, $\mathcal{P}(N)$ being a state

To simplify, it means that at each transition, a node among the network will ask one neighbor about a possible routing, and add the route to the corresponding routing table if:

- the new route isn't the node itself: $X \neq Z$
- the new route doesn't pass through the node itself
- the node didn't know the route before: $r(z) \notin \chi$. This condition is not needed in the equation but it helps understanding the algorithm.

The final stable state occurs when all the nodes know all possible routes. An incomplete routing table means the global state isn't final.

Let us take for example the initial state of a diamond network topology with four nodes expressed in Maude:

```
n(1, v(2, r(2)) : v(3, r(3))) ;
n(2, v(1, r(1)) : v(4, r(4))) ;
n(3, v(1, r(1)) : v(4, r(4))) ;
n(4, v(2, r(2)) : v(3, r(3)))
```

We then launch the rewriting of the equation with Maude, and we find only one possible final state:

```
n(1, v(2, r(2) + r(3) + r(4)) :
  v(3, r(2) + r(3) + r(4))) ;
n(2, v(1, r(1) + r(3) + r(4)) :
  v(4, r(1) + r(3) + r(4))) ;
```

```
n(3, v(1, r(1) + r(2) + r(4)) :
  v(4, r(1) + r(2) + r(4))) ;
n(4, v(2, r(1) + r(2) + r(3)) :
  v(3, r(1) + r(2) + r(3)))
```

Each node knows all possible routes from its neighbors.

We tested several possible configurations, with some malicious nodes. Each time the final configuration allowed the nodes to know all the possible routes. You can find the Maude code as well as different examples at https://github.com/thomasarmel/qkd_dynamic_routing_protocol/.

6 CONCLUSION

Our protocol proposal responds to the problem addressed in the ETSI QKD standard regarding the transmission of keys in a QKD network, when all the KMEs would not be directly linked together via a quantum link. We freed ourselves from the assumption that "*each Trusted Node is securely operated and managed*", by allowing a minority of nodes on each routing layer to be compromised by an attacker without compromising confidentiality and authenticity of the secret. If there is a bottleneck in the network, then it is still necessary to trust the nodes that constitute the single point of passage. It is also possible for each node to adapt the secret sharing threshold according to the trust it puts on the nodes of the following layer. Since Shamir's Secret Sharing Scheme (*SSSS*) is information-theoretic safe, our protocol would retain its safety properties even if the attacker had access to quantum computation power.

Our protocol, however, has the disadvantage of increasing the number of messages sent in total, especially if the trust placed in the KMEs network is low.

In the case where the network becomes too large for its topology to be known in advance by all nodes, we then propose an algorithm allowing nodes to discover on the fly the routes between the node wanting to transmit the secret and the destination node. We have also made a formal analysis of the security of this on-the-fly route discovery algorithm.

REFERENCES

Bhatia, V. and Ramkumar, K. (2020). An efficient quantum computing technique for cracking RSA using Shor's algorithm. In *2020 5th ICCCA*, pages 89–94. IEEE.

Blakley, G. R. (1979). Safeguarding cryptographic keys. In *Managing Requirements Knowledge, Int. Workshop on*, pages 313–313. IEEE.

Blanchet, B. (2012). Automatic verification of security protocols in the symbolic model: The verifier proverif. In *Int. School on Foundations of Security Analysis and Design*, pages 54–87. Springer.

Blanchet, B. et al. (2016). Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135.

Blanchet, B., Smyth, B., Cheval, V., and Sylvestre, M. (2018). Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial. *Version from*, pages 05–16.

Choi, T., Kim, H., Kim, J., Yoon, C. S., and Lee, G. M. (2021). Quantum key distribution networks for trusted 5g and beyond: An ITU-T standardization perspective. In *2021 ITU Kaleidoscope: Connecting Physical and Virtual Worlds (ITU K)*. IEEE.

Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., and Talcott, C. (2003). The maude 2.0 system. In *Int. Conf. on Rewriting Techniques and Applications*. Springer.

Corniaux, C. L. and Ghodosi, H. (2014). An entropy-based demonstration of the security of Shamir's secret sharing scheme. In *2014 Int. Conf. on Information Science, Electronics and Electrical Engineering*, volume 1, pages 46–48. IEEE.

Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Trans. on IT*, 29(2):198–208.

ETSI, G. (2019). 014. Quantum Key Distribution (QKD); protocol and data format of REST-based key delivery API.

Kaluderovic, N. (2022). Attacks on some post-quantum cryptographic protocols: The case of the Legendre PRF and SIKE. Technical report, EPFL.

Mehic, M., Niemiec, M., Rass, S., Ma, J., Peev, M., Aguado, A., Martin, V., Schauer, S., Poppe, A., Pacher, C., et al. (2020). Quantum key Distribution: a networking perspective. *ACM Computing Surveys*.

Prévost, T., Martin, B., and Alibart, O. (2024). Formal verification of the ETSI proposal on a standard QKD protocol. *GT MFS 2024*.

Salvail, L., Peev, M., Diamanti, E., Alléaume, R., Lütkenhaus, N., and Länger, T. (2010). Security of trusted repeater quantum key distribution networks. *Journal of Computer Security*.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

Vyas, N. and Mendes, P. (2024). Relaxing trust assumptions on Quantum Key Distribution Networks. *arXiv preprint arXiv:2402.13136*.

Wang, M., Li, J., Xue, K., Li, R., Yu, N., Li, Y., Liu, Y., Sun, Q., and Lu, J. (2023). A segment-based multipath distribution method in partially-trusted relay quantum networks. *IEEE Communications Magazine*.

Zygelman, B. and Zygelman, B. (2018). No-cloning theorem, quantum teleportation and spooky correlations. *A First Introduction to Quantum Computing and Information*, pages 125–147.