Post-Quantum Secure Channel Protocols for eSIMs: Design, Validation and Performance Analysis

Luk Bettale[1][a], Emmanuelle Dottax[2][b] and Laurent Grémy[2][c]
[1]*IDEMIA Secure Transactions, Courbevoie, France*
[2]*IDEMIA Secure Transactions, Pessac, France*

Keywords: Post-Quantum Cryptography, Secure Channel, Protocol Design, Formal Proof, Embedded Device.

Abstract: The transition to Post-Quantum (PQ) cryptography is increasingly mandated by national agencies and organizations, often involving a phase where classical and PQ primitives are combined into hybrid solutions. In this context, existing protocols must be adapted to ensure quantum resistance while maintaining their security goals. These adaptations can significantly impact performance, particularly on embedded devices. In this article, we focus on standardized protocols which support application management on eSIMs across different modes. This is a complex use-case, involving constrained devices with stringent security requirements. We present PQ adaptations, including both hybrid and fully PQ versions, for all modes. Using ProVerif, we provide automated proofs that verify the security of these PQ variants. Additionally, we analyze the performance impact of implementing PQ protocols on devices, measuring runtime and bandwidth consumption. Our findings highlight the resource overhead associated with achieving post-quantum security for eSIM management.

1 INTRODUCTION

With quantum computing advances, the security of widely used cryptographic systems is increasingly threatened, making it essential to plan for migration to quantum-resistant solutions. With NIST's recent publication of Post-Quantum Cryptography (PQC) standards, transitions can now be planned. This is particularly urgent for sectors deploying long-lived equipment, such as automotive systems, smart meters, and critical infrastructure sensors. In these cases, devices remain operational for many years and rely on connectivity based on an embedded Subscriber Identity Module (eSIM), a digital version of a SIM card embedded directly within the device hardware. Securing these devices against quantum attacks is crucial, given the high cost of upgrading hardware in the field.

The migration of embedded elements and their associated remote management protocols is therefore critical. This work focuses on protocols for remote management of eSIMs (and, more generally, embedded secure elements), as defined by Global System for Mobile communications Association (GSMA) and GlobalPlatform (GP) standards. Specifically, we study a protocol supporting a *scripting* mode, enabling pre-generated, sequenced commands in environments without a continuous connectivity. This protocol makes use of Elliptic Curve Diffie-Hellman (ECDH) as a Non-Interactive Key Exchange (NIKE, (Freire et al., 2012)), and thus presents complexities for post-quantum migration. Indeed, no suitable PQ NIKE currently exists. Similar challenges are noted in Signal's key exchange (Brendel et al., 2020; Hashimoto et al., 2021; Brendel et al., 2022; Kret and Schmidte, 2024), WireGuard (Hülsing et al., 2021), TLS (Celi et al., 2022) or Noise (Angel et al., 2022).

We propose a PQ adaptation using only standardized PQ primitives. To our knowledge, this is the first approach combining a digital signature and key encapsulation for mutual authentication. Though designed for a specific use case, it may have broader applications. This migration also considers resource constraints in secure embedded elements, making it a valuable case study.

This paper presents the background and the target protocols in Sect. 2, followed by PQ and hybrid adaptations in Sect. 3, retaining their original security properties. Formal verification with ProVerif is detailed in Sect. 4. Finally, Sect. 5 evaluates the performance impact of migrating to PQ mechanisms.

[a] https://orcid.org/0000-0002-8799-8568
[b] https://orcid.org/0000-0001-8717-3462
[c] https://orcid.org/0009-0008-2715-3169

Bettale, L., Dottax, E. and Grémy, L.
Post-Quantum Secure Channel Protocols for eSIMs: Design, Validation and Performance Analysis.
DOI: 10.5220/0013507200003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 108-119
ISBN: 978-989-758-760-3; ISSN: 2184-7711

2 PROTOCOLS FOR ESIM

2.1 Context

The eSIM is a technological evolution of the traditional SIM card. Unlike physical SIM cards, eSIMs are programmable secure chips soldered directly onto the device's motherboard[1]. A further evolution is the Integrated SIM (iSIM), which is incorporated into the device's processor. Both technologies enable the management of mobile subscriptions and various services without requiring physical intervention.

The GSMA defines the interactions between the various involved stakeholders, including Mobile Network Operators (MNOs), Original Equipment Manufacturers, and eSIM providers, across different use-cases. For instance, in the process of downloading a SIM *profile* (when a user acquires a new subscription), the GSMA specifies how the new profile is transferred from the MNO to the eSIM, via the mobile device (GSMA, 2023a; GSMA, 2023b).

Since eSIMs function as secure elements, they can host additional security-sensitive services. For instance, a banking mobile application could benefit from a counterpart on the eSIM for secure operations. In these scenarios, the involved parties differ: indeed, the *service provider*, as owner of the application, enters in the process. A distinct set of documents governs these cases: the *Secured Applications for Mobile* specifications (GSMA, 2023c) outlines how applications can be installed and managed on eSIMs. A secure link between the eSIM and the entity responsible for installation is established using the *Secure Channel Protocol '11'* (SCP11). This protocol is specified by GP, the organization in charge of standards for digital services that rely on secure elements (GlobalPlatform, 2023). This document specifies a secure channel protocol based on Elliptic Curve Cryptography (ECC). We explore it in more detail in the next section.

2.2 Protocol of GP SCP11

SCP11 is a secure channel protocol widely used in secure element-based products, including eSIMs. It employs ECC for mutual authentication and secure channel initiation, and AES for transport encryption. It defines three variants (referred to as *modes*), each tailored to a specific use case. The protocol involves two parties: the Card (e.g., an eSIM) and the Off-Card Entity (OCE, such as a terminal or server). Table 1 summarizes the security properties claimed in (GlobalPlatform, 2023, §4.3-4.6), including authentication, confidentiality, integrity, Perfect Forward Secrecy (PFS) and session replay (these properties are discussed in Sect. 4). The different modes are detailed below.

Table 1: Properties of the different SCP11 modes.

Property	Mode A	Mode B	Mode C
Authentication OCE to Card	✔		✔
Authentication Card to OCE	✔	✔	✔
Message Integrity	✔	✔	✔
Data Confidentiality	✔	✔	✔
Perfect Forward Secrecy	✔	✔	
Session Replay (replayable)			✔

2.2.1 SCP11 Mode A

Mode A enables mutual authentication between the Card and the OCE, as illustrated in Fig. 1. Both parties hold an ECC key pair—$(\mathsf{EC.sk_{OCE}}, \mathsf{EC.pk_{OCE}})$ for the OCE and $(\mathsf{EC.sk_C}, \mathsf{EC.pk_C})$ for the Card—along with associated certificates $\mathsf{Cert_{OCE}}$ and $\mathsf{Cert_C}$ signed by an authority CA. Authentication is achieved via static key ECDH, while a second, ephemeral ECDH ensures PFS. The process involves the OCE sending *commands* to trigger computations on the Card, which responds accordingly.

The process begins with the OCE retrieving the Card's certificate through a GET_DATA command and verifying it using the CA's public key $\mathsf{EC.pk_{CA}}$. The OCE then sends its own certificate via a Perform Security Operation (PSO) command, which the Card verifies. Next, the OCE generates an ephemeral key pair $(\mathsf{EC.epk_{OCE}}, \mathsf{EC.esk_{OCE}})$ and sends the public component to the Card through a MUTUAL_AUTH command. The Card, in turn, creates its own ephemeral key pair $(\mathsf{EC.epk_C}, \mathsf{EC.esk_C})$ and performs two elliptic curve key agreements ($\mathsf{EC.KeyAgr}$, standard ECDH as specified in (BSI, 2018, §4.3)): one using the certified keys and another with the ephemeral keys, ensuring PFS.

The shared secrets, Sss and See, are processed through a key derivation function KDeriv to generate a receipt key $\mathsf{SK_{Receipt}}$ and session keys for secure channel $\mathsf{SK_{Session}}$[2]. The KDeriv function uses SHA-256, as specified in (BSI, 2018). The message $\mathsf{Receipt}$ is an AES-CMAC (Dworkin, 2005) computed over

[1] Strictly speaking, eSIM refers to the service, while eUICC (Embedded Universal Integrated Circuit Card) denotes the physical device operating the mobile functionality. However, these terms are frequently used interchangeably.

[2] In the specification (GlobalPlatform, 2023, Table 6.18), $\mathsf{SK_{Receipt}}$ is the *receipt key*, while $\mathsf{SK_{Session}}$ comprises the S-ENC, S-MAC, S-RMAC and S-DEK keys.

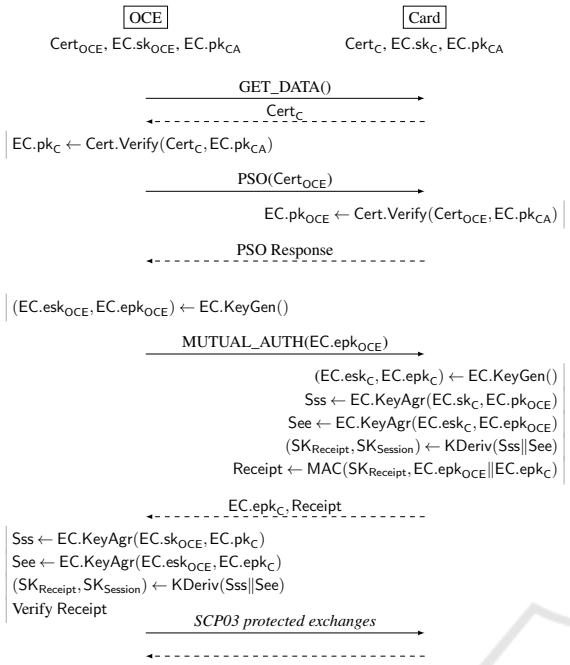

Figure 1: SCP11 Mode A.

$EC.epk_{OCE}$ and $EC.epk_C$, and sent to the OCE along with the Card's ephemeral key. This enables the OCE to compute and verify the MAC value, allowing it to authenticate the Card before initiating the secure channel.

If verified, the OCE and the Card continue with the *Secure Channel Protocol '03'* (SCP03) (GlobalPlatform, 2020), using $SK_{Session}$, which includes one encryption/decryption key and two MAC keys, one for each direction. All messages are encrypted and MAC-ed.

It can be noted that the Card cannot authenticate the OCE until it receives an SCP03 command with a valid MAC, as prior messages could come from an attacker without access to $EC.sk_{OCE}$. Additionally, the session cannot be replayed.

2.2.2 SCP11 Mode B

The Mode B provides authentication of the Card to the OCE only; it is illustrated by Fig. 2. This mode is useful when the OCE lacks certified keys, but can achieve a limited level of authentication through alternative means. For instance, if the OCE is a card terminal, a user-entered PIN verified by the Card offers a weak, indirect form of OCE authentication. However, these considerations are out of the scope of (GlobalPlatform, 2023), and are not discussed further here.

Mode B is similar to Mode A, with two distinctions:

- The OCE does not send a certificate or use static keys (no PSO command is sent);
- The two key agreements involve the OCE's ephemeral key and both the static and ephemeral keys of the Card (the command INTERNAL_AUTH replaces the MUTUAL_AUTH one).

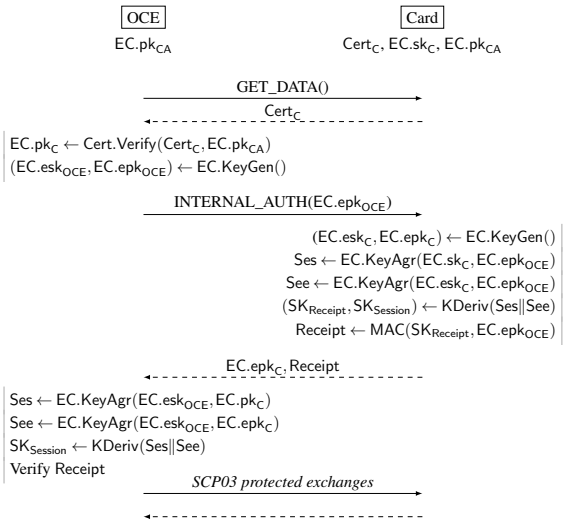

Figure 2: SCP11 Mode B.

The rest of the protocol remains the same, ensuring PFS due to the use of ephemeral keys, and anti-replay.

2.2.3 SCP11 Mode C

Mode C supports *offline scripting*, allowing the OCE to pre-generate commands for later execution by a third-party, as shown in Fig. 3. Typically, this mode assumes the Card is an eSIM in a mobile device, which plays the script. For instance, the script might be distributed as part of a rich environment application (e.g., an Android or iOS application) and executed during the application installation. Importantly, no secrets are handled by the mobile application, enabling the safe installation of services on the eSIM to secure functions such as those of a banking application. This mode also allows for remote administration of secure elements in general, without requiring a continuous, direct connection, which may be impractical in certain scenarios.

Mode C resembles Mode A with the following modifications:

- The OCE retrieves the eSIM's public key from a database.
- Only the eSIM's static keys are used, as the *offline* nature of this mode precludes the use of ephemeral data on eSIM side.

Mutual authentication, key derivation and command

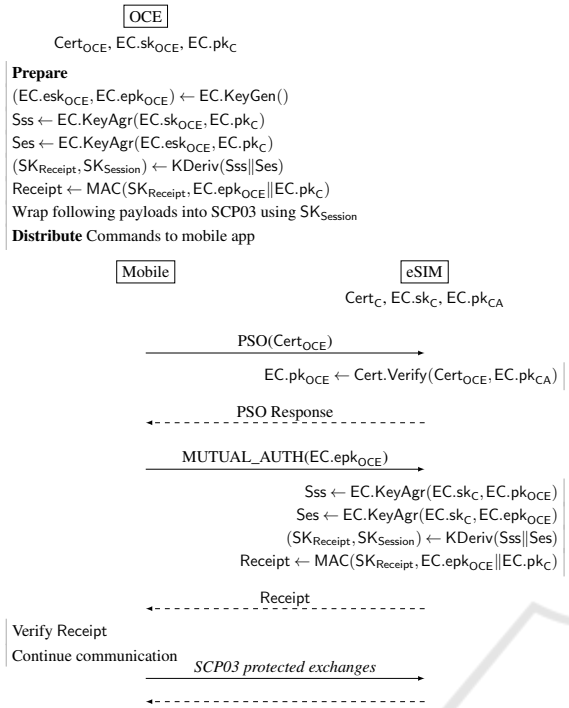

Figure 3: SCP11 Mode C.

wrapping are performed by the OCE in advance. The Receipt is based on $\mathsf{EC.epk_{OCE}}$ and $\mathsf{EC.pk_C}$.

Mode C provides mutual authentication between the OCE and the eSIM but lacks PFS for the eSIM's secrets. Furthermore, as the session relies only on static data from the eSIM, it can be replayed, unlike in Modes A and B. This is acknowledged by (GlobalPlatform, 2023), and as a consequence some sensitive commands—like inserting new keys—are disallowed in Mode C.

3 QUANTUM-RESISTANT VERSIONS

As seen, current protocols rely on ECC, which is vulnerable to quantum attacks, necessitating a shift to quantum-resistant alternatives. This is crucial to protect long-term sensitive data and secure eSIMs in devices with extended lifespans, like vehicles or smart meters. This section addresses the complexities of this migration and the required adaptation efforts.

3.1 Quantum-Resistant Mode C

We analyze Mode C, the variant used for scripting, where the off-card entity (OCE) prepares a script for the eSIM. Mutual authentication ensures that only the intended eSIM, with the correct ECDH key, can read the script, while the eSIM can verify its origin. Our goal is to maintain these properties against quantum-capable adversaries.

Standardized post-quantum algorithms now exist for signature (NIST, 2024a; NIST, 2024c) and Key Encapsulation Mechanism (KEM) (NIST, 2024b). Since the original protocol relies on ECDH, a natural approach is replacing it with a KEM. However, this alone fails: while the OCE can encrypt for the eSIM using its long-term KEM public key, it cannot prove authorship of the script, as KEMs require interaction for authentication. This limitation is also observed in, e.g., post-quantum adaptations of Signal (see Sect. 1): while KEMs work for key establishment, they cannot replace ECDH in protocols requiring asynchronous authentication.

To address this, the OCE needs a signature key to enable the eSIM to verify the script's origin. Replacing the eSIM's long-term KEM key with a signature key is not feasible here, as it would again require interaction with the eSIM, allowing it to sign an ephemeral KEM key.

Thus, we construct a protocol with a signature key on the OCE side, and a KEM key on the eSIM side. To the best of our knowledge, this is the first approach of its kind to leverage such a combination. The protocol is illustrated by Fig. 4 (the colored text is for a hybrid version presented in Sect. 3.3 and should be omitted for now). The OCE is now equipped with a signature key $\mathsf{SIG.sk_{OCE}}$ and the corresponding certificate $\mathsf{Cert_{OCE}}$. As in the classic case, the OCE already knows the eSIM's public key, which is now a KEM key $\mathsf{KEM.pk_C}$. The OCE begins by performing an encapsulation on this key to produce a shared secret Ss and its ciphertext c.Ss, which are used to derive the secure channel and receipt keys. The ciphertext is signed, to authenticate the origin of the script. The script is sent to the mobile device and played on the eSIM. The eSIM verifies the OCE's certificate, checks the signature, and decapsulates the ciphertext to obtain the shared secret and derive the session keys. The remainder of the process aligns with the classic protocol.

Regarding PFS, the situation is the same as in the classical case: it is not ensured on the eSIM's side (recovering $\mathsf{KEM.sk_C}$ allows to decrypt past communications), but it is on the OCE's one (the shared secret Ss is ephemeral). Similarly, replay is possible.

3.2 Quantum-Resistant Modes A and B

For completeness and coherence, we explore how PQ versions of Modes A and B can be constructed using

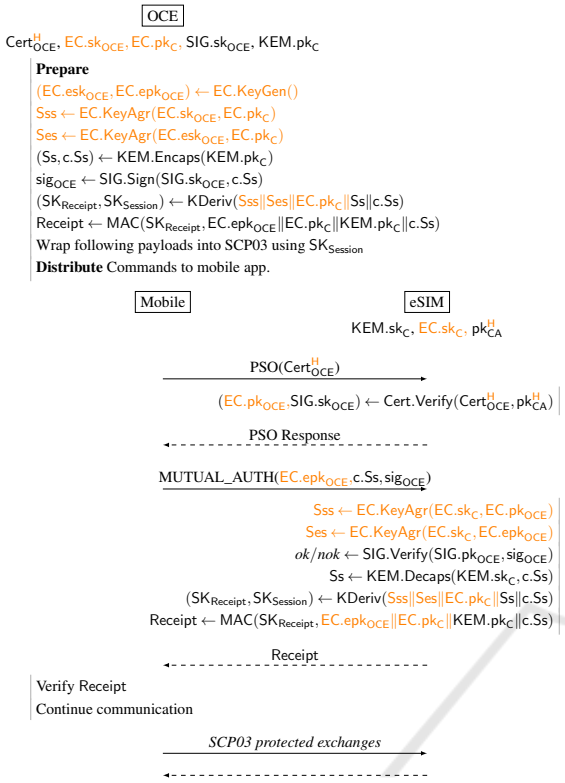

Figure 4: Hybrid version for Mode C (the colored part can be omitted for a purely post-quantum version).

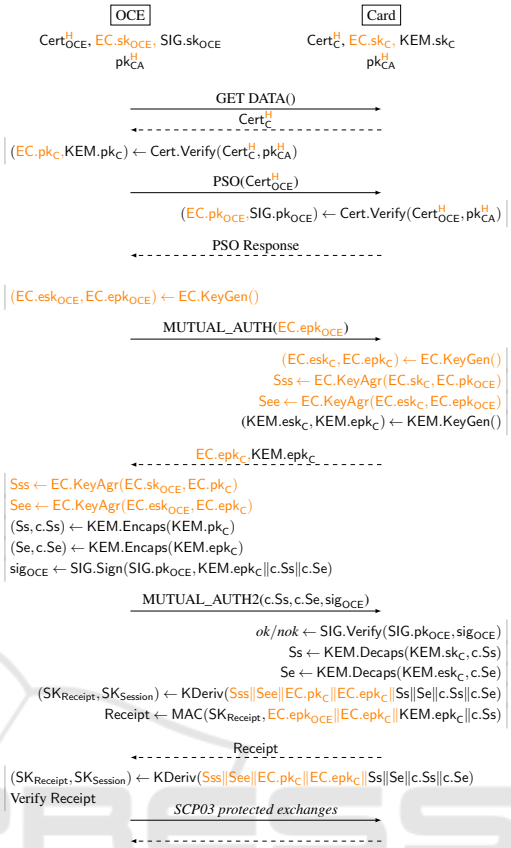

Figure 5: Hybrid version for Mode A (the colored part can be omitted for a purely post-quantum version).

the same credentials as Mode C: a signature key for the OCE and a KEM key for the eSIM. While these protocols could certainly rely exclusively on either signature keys or KEM keys—and possibly achieve better performances in these setting—using the same functions as in Mode C would streamline the process. This approach removes the need for additional credentials and avoids introducing extra functionalities on the eSIM.

For Mode A, we want to maintain the original mutual authentication and PFS properties. This can be achieved following the process illustrated by Fig. 5 (ignoring the colored part for now). Compared to the classic protocol, we need one more command, that we call MUTUAL_AUTH2. This is again due to the usage of a KEM in place of an ECDH key agreement.

The beginning of the protocol is the same: the Card and the OCE exchange their certificates, verify them using $\mathsf{SIG.pk_{CA}}$ and extract the public keys $\mathsf{SIG.pk_{OCE}}$ and $\mathsf{KEM.pk_C}$. Of course, the certificates are signed with a PQ signature algorithm. The first MUTUAL_AUTH command is used to make the eSIM generate a KEM ephemeral key and send the public part to the OCE. The OCE can then use this ephemeral key and the static one to generate two shared secrets Se and Ss, and their respective ciphertexts c.Se and c.Ss. The OCE signs these ciphertexts, together with the eSIM's ephemeral key. The resulting signature and the ciphertexts are sent to the eSIM via the new command. The eSIM can then verify the signature, decapsulate both ciphertexts and derive secret material from the concatenation of shared secrets and ciphertexts (as advised by (ETSI, 2020)). The rest of the protocol is unchanged: the eSIM computes an authentication value Receipt, and both parties use the session keys $\mathsf{SK_{Session}}$ for further exchanges through SCP03.

Note that the ephemeral key could be generated by the OCE, this is equivalent for the PFS (assuming both parties correctly manage ephemeral values), and both options need two commands for the mutual authentication.

The PQ version of Mode B is similar to the one of Mode A, with the difference that the OCE does not send any certificate, nor perform any signature. It is illustrated by Fig. 6 (again ignoring the colored part).

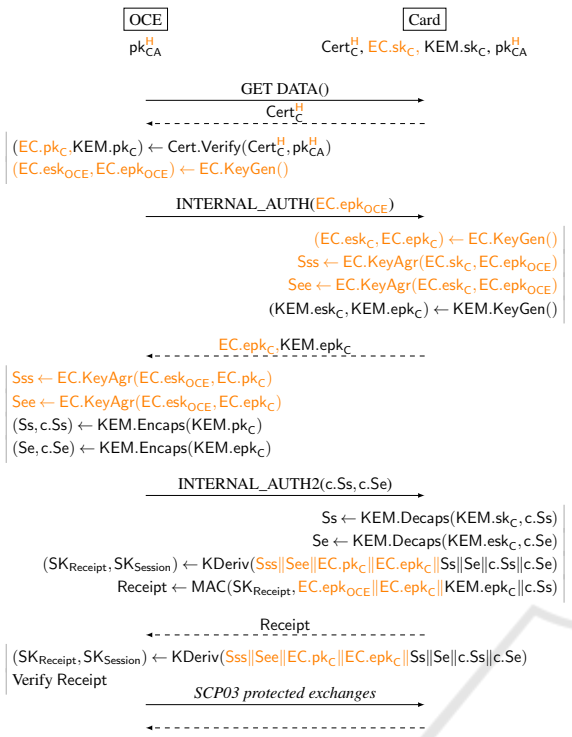

Figure 6: Hybrid version for Mode B (the colored part can be omitted for a purely post-quantum version).

3.3 Hybrid Versions

Hybrid versions combine classical and PQ algorithms to mitigate the limited maturity of PQ solutions. This ensures security as long as one algorithm set remains secure. This approach is endorsed by European cybersecurity agencies (ANSSI, 2023; BSI, 2024) and supported by NIST (Alagic et al., 2025).

Several options are available for constructing hybrid protocols, the main ones are:

- *Concatenate* hybrid key agreement, where both cryptographic methods are combined in single messages by concatenating their respective data.
- *Cascade* hybrid key agreement, where the first key agreement is executed before the second.

With the first option, each command conveys more data, while the second one augments the number of commands. We opted for the first option—the second one can easily be derived.

Figure 5 illustrates the hybrid protocol for Mode A, the classical part being colored. In this set-up, the OCE and the eSIM use hybrid certificates, denoted $\mathsf{Cert}^{\mathsf{H}}_{\mathsf{OCE}}$ and $\mathsf{Cert}^{\mathsf{H}}_{\mathsf{C}}$. Multiple formats for hybrid certificates exist (see (Gray and Onsworth, 2024) for an overview), but all include both classical and PQ keys and are signed by the classical and PQ keys of the CA, denoted $\mathsf{pk}^{\mathsf{H}}_{\mathsf{CA}}$. The Cert.Verify function returns both classical and PQ public keys: for the OCE, $\mathsf{EC.pk}_{\mathsf{OCE}}$ and $\mathsf{SIG.pk}_{\mathsf{OCE}}$; for the eSIM, $\mathsf{EC.pk}_{\mathsf{C}}$ and $\mathsf{KEM.pk}_{\mathsf{C}}$. Hybrid versions of Modes B and C (Fig. 6 and 4) are based on these same principles.

Our hybrid protocols incorporate both classical and PQ computations while maintaining the same command count as the PQ-only versions. Shared secrets are derived from classical and PQ key agreements, then combined using the KDeriv function. This step requires careful design to ensure hybrid security. This aspect has been studied in various works (Herzberg, 2009; Giacon et al., 2018; Bindel et al., 2019). Currently, only a few standardized options exist (ETSI, 2020; Alagic et al., 2025). Following these standards, we input KDeriv with all shared secrets, along with the public keys for ECDH and the ciphertexts for KEM.

As seen, implementing these new protocols requires modifying the commands sent to the eSIM. The approach presented is one possibility, but the exact implementation will be defined by GlobalPlatform.

4 FORMAL PROOFS

The security properties of the different modes of SCP11 are listed in Sect. 2.2 and summarized in Table 1. However, no formal security proof exists in the literature confirming that these properties are achieved. We therefore propose a formal verification of the different modes of SCP11, and then consider the PQ-only and hybrid variants.

4.1 Symbolic Formal Verification

Our proofs are conducted in the symbolic model, also known as the Dolev–Yao attacker model (Dolev and Yao, 1983). In this setting, the protocol participants communicate over a public channel and employ *perfect* cryptographic primitives, where, for instance, decryption is only possible with the corresponding key. An active attacker in this model has full control over the public channel, enabling her to read, drop, modify, replay messages, and inject new messages. Additionally, the attacker may selectively compromise participants, e.g., through long term key leaks, or gain access to oracles that break cryptographic assumptions, e.g., retrieving a private key from a public key. In this setting, we can model a protocol and determine whether certain (mathematical) properties hold despite such an adversary. We designate the OCE, the Card and the attacker as $\mathcal{O}$, $\mathcal{C}$ and $\mathcal{A}$, respectively.

Authentication. Formalizing cryptographic properties into precise mathematical formulations can be challenging, as illustrated by the various definitions of authentication (Lowe, 1997). Interested readers can find a near-mathematical formalization of these properties in (The Tamarin Team, 2024, pp.82–83). We aim to prove the two strongest forms of agreement for classical SCP11: the *non-injective agreement* (Lowe, 1997, §2.3) and *(injective) agreement* (Lowe, 1997, §2.4). The authentication of $\mathcal{C}$ to $\mathcal{O}$ may represent an injective agreement upon Receipt. The authentication of $\mathcal{O}$ to $\mathcal{C}$, which is unattainable in Mode B, may represent an injective agreement on the SCP commands in Mode A, but a non-injective agreement in Mode C, since $\mathcal{O}$ may replay a previous session.

Message Integrity. Message integrity is required during SCP03 exchanges. For any command a sent by $\mathcal{O}$ and any command b received by $\mathcal{C}$, integrity is achieved if a and b are identical. However, this condition may not hold for the Mode B, where any $\mathcal{A}$ may impersonate an $\mathcal{O}$. Message integrity is verified across all modes if, for any command a whose authenticated encryption e is sent by $\mathcal{O}$ and for any command b decrypted after the reception of e by $\mathcal{C}$, the commands a and b are the same.

Data Confidentiality and PFS. Like message integrity, data confidentiality applies only to SCP03 exchanges. A command is confidential if $\mathcal{A}$ cannot access it at any point. Proving PFS adds a temporal condition: we assume $\mathcal{A}$ obtains the long-term secret key of $\mathcal{C}$ or $\mathcal{O}$ after the protocol's completion. A protocol is perfect forward secure (Günther, 1990; Diffie et al., 1992) if $\mathcal{A}$ cannot compute the symmetric keys, such as $\mathsf{SK}_{\mathsf{Session}}$, exchanged before impersonating $\mathcal{C}$ or $\mathcal{O}$.

Session Replay. The session replay is also addressed by the (non-)injection agreement discussed above. To validate the negation of this property, session uniqueness, we need to confirm that if two tuples of keys ($\mathsf{SK}_{\mathsf{Receipt}}$, $\mathsf{SK}_{\mathsf{Session}}$) established by $\mathcal{C}$ and $\mathcal{O}$ are identical at two different time periods, those periods must be the same.

To be automatically proved, all these (mathematical) properties need to be encoded in the formalism of software programs designed for the formal verification of cryptographic protocols. The two leading tools for formally proving protocols in the symbolic model are ProVerif (Blanchet et al., 2008) and Tamarin (Schmidt et al., 2012). A comprehensive comparison of these tools, along with others, can be found in (Barbosa et al., 2021a, §II.A-C). Two newer software options may also be considered: Verifpal (Kobeissi et al., 2020) and PQ-Squirrel (Cremers et al., 2022). All these tools generally involve two stages: the first models the protocol, and the second assesses the security properties, expressed as (mathematical) queries. In this work, we chose to use ProVerif (version 2.05).

4.2 ProVerif Security Protocol Analysis

In this section, we discuss our modeling choices in the ProVerif language. For comprehensive details, refer to the ProVerif manual (Blanchet et al., 2023).

4.2.1 Model

ProVerif requires explicit definitions of the cryptographic primitives used in the protocol. These definitions resemble an API description, accompanied by the mathematical equations each primitive verifies (Blanchet et al., 2023, §4.2). For instance, a message authentication code (MAC) is illustrated in Listing 1. To compute a MAC, the function Mac takes a key k and a message m to produce the MAC c of m. To verify if c is the MAC of m under the key k, the function MacVerify takes the key k, the message m and the alleged MAC c and outputs *true* if $c = \mathrm{Mac}(k,m)$ (i.e., $\mathrm{MacVerify}(k,m,\mathrm{Mac}(k,m)) = 1$) and *false* otherwise (i.e., $\mathrm{MacVerify}(k,m,c \neq \mathrm{Mac}(k,m)) = 0$).

Listing 1: ProVerif model of the MAC computation and verification.

```
type mackey. type macs.
fun Mac(mackey, bitstring): macs.
fun MacVerify(mackey, bitstring, macs): bool
reduc forall k: mackey, m: bitstring;
  MacVerify(k, m, Mac(k, m)) = true
otherwise forall k: mackey, m: bitstring, c: macs;
  MacVerify(k, m, c) = false.
```

The primitives are embedded within two *process macros* `pCard` and `pOCE` representing the Card and the OCE in Listing 3. Unlike Tamarin which relies on state sharing, we define a single process macro for each actor. The two actors communicate over an open channel, vulnerable to a Dolev–Yao attacker (see Sect. 4.1). Each process macro is initialized in the main process with the public and private keys required for the chosen mode, with public keys also broadcasted over the public channel, making them accessible to the attacker. Additionally, the certificate authority is modeled as a separate macro process that signs trusted public keys stored in an immutable table.

4.2.2 Queries

To prove PFS, ProVerif allows to work with the notion of *phase* (Blanchet et al., 2023, §4.1.6). The complete execution of the protocol is considered phase 0, followed by phase 1, which necessarily occurs after the completion of phase 0, during which the long term keys are leaked. The proof of PFS is done for phase 1.

The confidentiality query is expressed in Listing 2, where we define a command and query whether the attacker obtains this command.

Listing 2: ProVerif model of a confidentiality query.

```
free commands: bitstring [private].
query attacker(commands).
```

To formalize other queries in the ProVerif model, we first create *events*. An *event* denotes a significant sequence of actions. For instance, the authentication of the Card to the OCE via injective agreement on the receipt message is briefly formalized in Listing 3. The issuance and sending of the receipt message by the Card to the OCE is the goal of `sendRec`, which takes as arguments the receipt message and the two actors, first the sender and second the receiver. To prove the authentication query, we must define another event to confirm that the MAC verification is correctly performed: this is the goal of `acceptRec`, which takes as arguments the receipt message and the two actors, first the receiver and second the issuer. Once events are defined and placed correctly in the model, we use the injective correspondence with the `inj-event` keyword.

Listing 3: ProVerif model of the two process macros.

```
...
event sendRec(macs, host, host).
event acceptRec(macs, host, host).
...
query rec: macs, OCE: host, Card: host,
  i: time, j: time;
  event(acceptRec(rec, OCE, Card))@i ==>
  inj-event(sendRec(rec, Card, OCE))@j &&
  j < i.
...
let pOCE(pkS: signCApub, SKOCE: eckapriv) =
  ...
  in(c, (ePKSD: eckapub, rec: macs));
  ...
  let (rkey: mackey, senc: symkey, smac: mackey) =
    kdf((ShSss, ShSee)) in
  if MacVerify(rkey, (ePKOCE, ePKSD), rec) then (
    event acceptRec(rec, OCE, Card);
    ...
  ).
let pCard(pkS: signCApub, SKSD: eckapriv) =
  ...
  let (rkey: mackey, senc: symkey, smac: mackey) =
    kdf((ShSss, ShSee)) in
  let rec = Mac(rkey, (ePKOCE, ePKSD)) in
  event sendRec(rec, Card, OCE);
  out(c, (ePKSD, rec));
  ...
```

Regarding the integrity query in Listing 4, events relate to the (encrypted) commands sent over the SCP03 channel. The query states that if an actor sends a command a through an encrypted message e with an authentication tag m, and another actor receives a command b from the decryption of message e with tag m, then a and b are the same.

Listing 4: ProVerif model of an integrity query.

```
event sendCom(bitstring, bitstring, macs).
event readCom(bitstring, bitstring, macs).
query a: bitstring, b: bitstring, e: bitstring,
  m: macs; event(sendCom(a, e, m)) &&
  event(readCom(b, e, m)) ==> a = b.
```

Finally, Listing 5 formalizes the session uniqueness query. After the Card derives the key, the event `SCP03SK` is triggered. This property states that if the three keys established by the Card with the (believed) OCE are identical for different time periods i and j, then i and j must match, i.e., $i = j$.

Listing 5: ProVerif model of a replay query.

```
event SCP03SK(mackey, symkey, mackey, host, host).
query rkey: mackey, senc: symkey, smac: mackey,
  i: time, j: time;
  event(SCP03SK(rkey, senc, smac, Card, OCE))@i &&
  event(SCP03SK(rkey, senc, smac, Card, OCE))@j
  ==> i = j.
```

4.3 Verification of the Models

We model all three modes. In our ProVerif models, we include a sanity check to verify that at least one trace allows the Card to receive the commands sent by the OCE. Without this check, a query may be considered as verified only because the event(s) of the query are not reached; for instance, if the first event is not reached, both true and false can be assumed for the second event, see Listing 3. All properties are verified in less than 3 seconds on a personal computer.

4.4 Post-Quantum Protocols

With the ProVerif models established for the classical modes of SCP11, we now aim to prove that the quantum-resistant versions introduced in Sect. 3 achieve the same properties as their classical counterparts in the symbolic model.

4.4.1 Previous Works

Unlike computational tools (Barbosa et al., 2021b; Blanchet and Jacomme, 2023), symbolic tools typically do not require adaption to take into account quantum adversaries. Consequently, we can utilize ProVerif directly, given an appropriate model for the new primitives introduced in the post-quantum variants. Several studies have already explored automated proofs in the symbolic model for post-quantum versions of various protocols, as summarized in Table 2.

Table 2: Post-quantum protocols with security proofs in the symbolic model with an automated tool.

(Sub)Protocol	Article	ProVerif	Tamarin	Verifpal	PQ-Squirrel
OPC UA	(Paul and Scheible, 2020)	✔		✔	
PQ-Wireguard	(Hülsing et al., 2021)		✔		
PQ IKEv2	(Gazdag et al., 2021)		✔		
KEMTLS	(Celi et al., 2022)		✔		
PQ IKEv1	(Cremers et al., 2022)				✔
PQ IKEv2	(Cremers et al., 2022)				✔
PQ X3DH	(Cremers et al., 2022)				✔
PQ Signal	(Beguinet et al., 2024)		✔		
PQXDH	(Bhargavan et al., 2024)	✔			
iMessage PQ3	(Linker et al., 2024)		✔		
PQ SCP11	This work	✔			

4.4.2 Modeling PQ SCP11

The main distinction from classical protocols is the introduction of a KEM algorithm. Following (Bhargavan et al., 2024), a generic KEM encapsulation is modeled as the generation of a secret value *m*, asymmetrically encrypted with the recipient's public key, where *m* serves as the shared secret. However, the ML-KEM standard (NIST, 2024b) differs by using a shared secret which can be modeled as (the hash of) *m* concatenated with the public key, as shown in Listing 6. Unless stated otherwise, all our proofs use a generic KEM model.

Listing 6: ProVerif model of a simplified version of ML-KEM.

```
type kempriv. type kempub.
fun kempk(kempriv): kempub.
fun h(bitstring, kempub): bitstring.
fun pkeenc(kempub, bitstring): bitstring.
fun pkedec(kempriv, bitstring): bitstring
reduc forall sk: kempriv, m: bitstring;
  pkedec(sk, pkeenc(kempk(sk), m)) = m.
letfun kemenc(pk: kempub) = new m: bitstring;
  let k = h(m, pk) in (pkeenc(pk, m), k).
letfun kemdec(sk: kempriv, ct: bitstring) =
  let m = pkedec(sk, ct) in h(m, kempk(sk)).
```

To prevent public key confusion attacks (Bhargavan et al., 2024, §4.1.1), we incorporate an identifier into public key certificates, ensuring domain separation. We model the PQ signature algorithm as the standard signature model (Jackson et al., 2019, §2.2).

PQ-Only versions. The events and queries from the classical models can be directly reused in the models for the PQ SCP11 versions. Our models for the PQ modes validate the expected proofs outlined for the classical versions, as summarized in Table 1. Notably, the PQ versions of Modes A and C allow to reach the authentication of the OCE to the Card sooner than in the classical version through the use of signatures. More precisely, the sequence involving the issuance of a signature by the OCE followed by its verification by the Card guarantees *aliveness* (Lowe, 1997, §2.1) of the OCE in Mode C. This same sequence guarantees to the Card *agreement* with the OCE on the signature in Mode A; however, the omission of the ephemeral key in the signed message reduces the authentication to *aliveness*, akin to Mode C. This reduction may be acceptable if the PQ protocol aims to align with the timeline the security properties achieved in the classical protocol.

During the modeling stage of the PQ protocols, we model firstly a simple rough key derivation step, by only considering the concatenation of the shared secret *without* taking into account the ciphertexts. This is in contrast with what is described for example in (ETSI, 2020, §8). While assessing the properties related to SCP11, we found that our initial (incorrect) model did not significantly diverge from expectations, except for the replay property in Mode B. In this case, since the OCE is not authenticated, an attacker could send the same shared secret, for two runs of the protocol. Interestingly, using ML-KEM (NIST, 2024b) defeats this attack since the public key used for the encapsulation is embedded into the shared secret construction, but this is not a perennial solution. By aligning our model on the protocol described in Fig. 6, the attack is defeated for a generic KEM.

Hybrid Versions. Hybrid protocols are of first importance today to prepare the migration between pure-classical and pure-post-quantum protocols. In addition to ensuring that an SCP11 mode provides the requisite security when its primitives are secure, hybrid protocols must also remain secure if either of the combined primitives—classical or else PQ—is compromised. To model this, we introduce oracles for the attacker via ProVerif *process macros*, allowing her to submit public keys and obtain the associated secret keys, but only for either PQ or classical primitives, as shown in Listing 7.

Listing 7: ProVerif model of the oracle which breaks the public key of a KEM.

```
free att: channel. (* Channel for the attacker *)
fun recoverKEMpriv(kempub): kempriv
reduc forall sk: kempriv;
```

```
  recoverKEMpriv(kempk(sk)) = sk [private].
let kem_attacks() = in(att, pk: kempub);
  out(att, recoverKEMpriv(pk)).
```

We can verify that the hybrid protocols ensure the same security properties as the classical protocols when PQ primitives are broken, and conversely the same security as PQ protocols when the classical primitives are compromised. Note that, since the signatures issued by the certificate authority are also hybrid, we model it as a single unbroken signature algorithm rather than a combination of two signature algorithms—one compromised and one secure.

5 IMPLEMENTATION

When implementing cryptographic protocols in embedded devices, a lot of constraints come into play. First, the amount of available resources (RAM, CPU frequency) is usually very low. Implementing post-quantum algorithms can be challenging. This is even more the case as implementations on embedded devices have to be resistant against physical attacks. The cost of securing an implementation against side-channel and fault attacks can drastically increase the required amount of RAM, as well as the execution time. On top of that, protocols are designed for two parties to communicate with each other. The amount of data exchanged by the parties would directly impact the overall performance of the protocols.

In this section, we study these metrics for an implementation on a typical embedded device (32-bit Cortex-M3 CPU at 100 MHz) with a hardware accelerator for ECC. The chip supports a baud-rate of 600 kbit/s. We discuss the impact of the choice of the PQ algorithm on the execution time of a complete protocol execution.

5.1 Algorithm Choice

For our experiments, we restrict ourselves to the algorithms that were standardized by the NIST, namely ML-KEM (NIST, 2024b) for key encapsulation, and ML-DSA (NIST, 2024a) and FN-DSA for signature. As the FN-DSA specifications are not yet available, we implemented the latest version submitted to the NIST competition, that is the Round 3 version of Falcon (Fouque et al., 2020). In the rest of the paper, we will use the name FN-DSA. The ML-KEM primitives were implemented with side-channel countermeasures such as (Coron et al., 2022; Coron et al., 2023).

We excluded SLH-DSA (NIST, 2024c) from our study because even with the *small* variant, the performances would be very low with processing time reaching up to 5 seconds whenever a verification is required. This algorithm is not suited for embedded devices.

To align the security levels with the post-quantum algorithms, we implemented the ECC component with three distinct security levels: 128, 192, and 256 bits, corresponding to NIST categories 1, 3, and 5, respectively. We used the `P-256`, `P-384` and `P-521` Elliptic Curve domain parameters as specified in SP800-186 (Chen et al., 2023). Since we simulated the OCE using a desktop computer, the timing results for the OCE are not relevant. However, the OCE typically has significantly greater processing power than the chip, making its timing negligible in this context.

5.2 Performance Analysis

In Tables 3 and 4, we present the communication and processing time of the chip for the hybrid version using the standardized post-quantum signatures, respectively ML-DSA and FN-DSA. Note that we measure only the secure channel establishment part, before any SCP03 exchanges. As the communication time may depend on the negotiated baud-rate and transmission protocol, it is clearly separated from the processing time of the chip. This communication time includes both the sending and receiving of data by the chip. Finally, we compare these implementations to a classic SCP11 using ECC by giving the ratio to see the overhead of hybridation.

For ML-DSA in Table 3, it is worth noticing that while the impact on the processing time is non-negligible, it is not more than a factor of 8 at worst. However, the communication time is more than 40 times slower. Despite this, the total time remains less than one second for the 128-bit security level.

The results for FN-DSA in Table 4 are much better due to the relatively small signature size and the efficient verification process. At the 128-bit security level, all modes run in less than 500 ms. In a context where the constrained device only performs verifications, it is advantageous to choose a signature algorithm with a short signature and fast verification, such as FN-DSA.

6 CONCLUSION

Through our exploration of SCP11's modes, we identify challenges in transitioning from classical to post-quantum cryptography. Beyond proving that SCP11 achieves its claimed security properties, we designed new PQ-only and hybrid protocols that uphold the

Table 3: Chip-based measurements of hybrid SCP11 protocols execution, with ratio relative to classic versions, using ML-DSA signature.

Protocol	Sec. level	Communication		Processing		Total	
		(ms)	(ratio)	(ms)	(ratio)	(ms)	(ratio)
Mode A	128	267	(×35)	516	(×7.7)	783	(×11)
	192	380	(×33)	796	(×6.1)	1176	(×8.3)
	256	515	(×32)	1184	(×5.3)	1698	(×7.1)
Mode B	128	117	(×23)	290	(×6.0)	406	(×7.6)
	192	165	(×21)	427	(×4.6)	591	(×5.9)
	256	228	(×22)	629	(×4.0)	857	(×5.1)
Mode C	128	167	(×43)	348	(×7.1)	516	(×9.8)
	192	239	(×42)	559	(×5.8)	797	(×7.8)
	256	321	(×40)	837	(×5.0)	1157	(×6.5)

Table 4: Chip-based measurements of hybrid SCP11 protocols execution, with ratio relative to classic versions, using FN-DSA (Round 3 Falcon) signature. Note that FN-DSA does not specify a parameter set for security level 192.

Protocol	Sec. level	Communication		Processing		Total	
		(ms)	(ratio)	(ms)	(ratio)	(ms)	(ratio)
Mode A	128	136	(×18)	341	(×5.1)	477	(×6.4)
	256	265	(×17)	750	(×3.3)	1014	(×4.2)
Mode B	128	82	(×16)	290	(×6.0)	371	(×7.0)
	256	162	(×15)	629	(×4.0)	791	(×4.7)
Mode C	128	72	(×19)	173	(×3.5)	244	(×4.6)
	256	137	(×17)	403	(×2.4)	540	(×3.0)

same guarantees as classical modes, validated by formal proofs.

We test different standardized PQ cryptographic primitives and FN-DSA, due to its short signatures and fast verification, proves to be the best fit. Even if FN-DSA signature generation is challenging to secure against side-channel attacks, there is no problem here as the card only needs to perform verification. Moreover, it is usually not possible to perform physical attacks on the OCE. Regarding performance on the card, it could be further enhanced through specialized accelerators, higher clock speeds, or expanded memory provided by chip manufacturers. Enhancing communication speed remains another avenue, though it requires coordinated efforts among stakeholders and will likely take longer to implement.

Future research will extend to other protocols in embedded ecosystems, offering formal proofs for PQ adaptations. Collaboration with GlobalPlatform working groups will support PQ standardization and ensure smooth migration across embedded applications.

ACKNOWLEDGEMENTS

The author would like to thank anonymous referees for useful and detailed comments on a previous version of the paper.

REFERENCES

Alagic, G., Barker, E., Chen, L., Moody, D., Robinson, A., Silberg, H., and Waller, N. (2025). *Recommendations for Key Encapsulation Mechanisms*. NIST. Special Publication 800-227, Initial Public Draft.

Angel, Y., Dowling, B., Hülsing, A., Schwabe, P., and Weber, F. (2022). Post quantum Noise. In *ACM CCS 2022*, page 97–109. ACM Press.

ANSSI (2023). *ANSSI views on the Post-Quantum Cryptography transition*.

Barbosa, M., Barthe, G., Bhargavan, K., Blanchet, B., Cremers, C., Liao, K., and Parno, B. (2021a). SoK: Computer-aided cryptography. In *2021 IEEE Symposium on Security and Privacy*, pages 777–795. IEEE Computer Society Press.

Barbosa, M., Barthe, G., Fan, X., Grégoire, B., Hung, S.-H., Katz, J., Strub, P.-Y., Wu, X., and Zhou, L. (2021b). EasyPQC: Verifying post-quantum cryptography. In Vigna, G. and Shi, E., editors, *ACM CCS 2021*, pages 2564–2586. ACM Press.

Beguinet, H., Chevalier, C., Ricosset, T., and Senet, H. (2024). Formal verification of a post-quantum Signal protocol with Tamarin. In Ben Hedia, B., Maleh, Y., and Krichen, M., editors, *VECoS '23*, volume 14368 of *LNCS*, pages 105–121. Springer.

Bhargavan, K., Jacomme, C., Kiefer, F., and Schmidt, R. (2024). Formal verification of the PQXDH post-quantum key agreement protocol for end-to-end secure messaging. In Balzarotti, D. and Xu, W., editors, *USENIX Security 2024*. USENIX Association.

Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., and Stebila, D. (2019). Hybrid key encapsulation mechanisms and authenticated key exchange. In Ding, J. and Steinwandt, R., editors, *PQCrypto 2019*, pages 206–226. Springer.

Blanchet, B., Abadi, M., and Fournet, C. (2008). Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51.

Blanchet, B. and Jacomme, C. (2023). CryptoVerif: a computationally-sound security protocol verifier. Technical Report RR-9526, Inria.

Blanchet, B., Smyth, B., Cheval, V., and Sylvestre, M. (2023). ProVerif 2.05: Automatic cryptographic protocol verifier, user manual and tutorial.

Brendel, J., Fiedler, R., Günther, F., Janson, C., and Stebila, D. (2022). Post-quantum asynchronous deniable key exchange and the Signal handshake. In Hanaoka, G., Shikata, J., and Watanabe, Y., editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 3–34. Springer.

Brendel, J., Fischlin, M., Günther, F., Janson, C., and Stebila, D. (2020). Towards post-quantum security for Signal's X3DH handshake. In Dunkelman, O., Jacobson, Jr., M. J., and O'Flynn, C., editors, *SAC 2020*, volume 12804 of *LNCS*, pages 404–430. Springer.

BSI (2018). *Elliptic Curve Cryptography*. Technical Guideline TR-03111, version 2.10.

BSI (2024). *Cryptographic Mechanisms: Recommendations and Key Lengths*. Technical Guideline TR-02102-1, version 2024-01.

Celi, S., Hoyland, J., Stebila, D., and Wiggers, T. (2022). A tale of two models: Formal verification of KEMTLS via Tamarin. In Atluri, V., Di Pietro, R., Jensen, C. D., and Meng, W., editors, *ESORICS 2022, Part III*, volume 13556 of *LNCS*, pages 63–83. Springer.

Chen, L., Moody, D., Randall, K., Regenscheid, A., and Robinson, A. (2023). *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*. NIST. Special Publication 800-186.

Coron, J.-S., Gérard, F., Montoya, S., and Zeitoun, R. (2022). High-order table-based conversion algorithms and masking lattice-based encryption. *IACR TCHES*, 2022(2):1–40.

Coron, J.-S., Gérard, F., Montoya, S., and Zeitoun, R. (2023). High-order polynomial comparison and masking lattice-based encryption. *IACR TCHES*, 2023(1):153–192.

Cremers, C., Fontaine, C., and Jacomme, C. (2022). A logic and an interactive prover for the computational post-quantum security of protocols. In *2022 IEEE Symposium on Security and Privacy*, pages 125–141. IEEE Computer Society Press.

Diffie, W., van Oorschot, P. C., and Wiener, M. J. (1992). Authentication and authenticated key exchanges. *DCC*, 2(2):107–125.

Dolev, D. and Yao, A. C. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207.

Dworkin, M. (2005). *Recommendation for Block Cipher – Modes of Operation: The CMAC Mode for Authentication*.

ETSI (2020). *Quantum-safe Hybrid Key Exchanges*. ETSI TS 103 744 V1.1.1.

Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. (2020). *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU*. specification v1.2.

Freire, E. S. V., Hofheinz, D., Kiltz, E., and Paterson, K. G. (2012). Non-interactive key exchange. Cryptology ePrint Archive, Paper 2012/732.

Gazdag, S.-L., Grundner-Culemann, S., Guggemos, T., Heider, T., and Loebenberger, D. (2021). A formal analysis of IKEv2's post-quantum extension. In *ACSAC '21*, page 91–105, New York, NY, USA. Association for Computing Machinery.

Giacon, F., Heuer, F., and Poettering, B. (2018). KEM combiners. In Abdalla, M. and Dahab, R., editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 190–218. Springer.

GlobalPlatform (2020). *Secure Channel Protocol '03' – Card Specification v2.3 — Amendment D – Version 1.2*.

GlobalPlatform (2023). *Secure Channel Protocol '11' – Card Specification v2.3 – Amendment F – Version 1.4*.

Gray, J. and Onsworth, M. (2024). Certificate mechanisms for transitioning to post-quantum cryptography. International Cryptographic Module Conference (ICMC) 2024.

GSMA (2023a). *RSP Architecture SGP.21 v3.1*.

GSMA (2023b). *RSP Technical Specification SGP.22 v3.1*.

GSMA (2023c). *Secured Applications for Mobile v1.1*.

Günther, C. G. (1990). An identity-based key-exchange protocol. In Quisquater, J.-J. and Vandewalle, J., editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 29–37. Springer.

Hashimoto, K., Katsumata, S., Kwiatkowski, K., and Prest, T. (2021). An efficient and generic construction for Signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In Garay, J., editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 410–440. Springer.

Herzberg, A. (2009). Folklore, practice and theory of robust combiners. *J. Comput. Secur.*, 17(2):159–189.

Hülsing, A., Ning, K.-C., Schwabe, P., Weber, F. J., and Zimmermann, P. R. (2021). Post-quantum WireGuard. In *2021 IEEE Symposium on Security and Privacy*, pages 304–321. IEEE Computer Society Press.

Jackson, D., Cremers, C., Cohn-Gordon, K., and Sasse, R. (2019). Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Cavallaro, L., Kinder, J., Wang, X., and Katz, J., editors, *ACM CCS 2019*, pages 2165–2180. ACM Press.

Kobeissi, N., Nicolas, G., and Tiwari, M. (2020). Verifpal: Cryptographic protocol analysis for the real world. In Bhargavan, K., Oswald, E., and Prabhakaran, M., editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 151–202. Springer.

Kret, E. and Schmidte, R. (2024). The PQXDH key agreement protocol. Technical report, Signal.

Linker, F., Sasse, R., and Basin, D. (2024). A formal analysis of Apple's iMessage PQ3 protocol. Cryptology ePrint Archive, Paper 2024/1395.

Lowe, G. (1997). A hierarchy of authentication specification. In *Computer Security Foundations Workshop*, pages 31–44. IEEE Computer Society.

NIST (2024a). *Module-Lattice-Based Digital Signature Standard*. FIPS 204.

NIST (2024b). *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. FIPS 203.

NIST (2024c). *Stateless Hash-Based Digital Signature Standard*. FIPS 205.

Paul, S. and Scheible, P. (2020). Towards post-quantum security for cyber-physical systems: Integrating PQC into industrial M2M communication. In Chen, L., Li, N., Liang, K., and Schneider, S. A., editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 295–316. Springer.

Schmidt, B., Meier, S., Cremers, C., and Basin, D. A. (2012). Automated analysis of Diffie-Hellman protocols and advanced security properties. In Chong, S., editor, *CSF 2012*, pages 78–94. IEEE Computer Society.

The Tamarin Team (2024). Tamarin prover manual.