Post-Quantum Protocols for Banking Applications

Luk Bettale[1], Marco De Oliveira[1], and Emmanuelle Dottax[2]

[1] IDEMIA, Courbevoie, France
[2] IDEMIA, Pessac, France
{luk.bettale,marco.deoliveira,emmanuelle.dottax}@idemia.com

Abstract. As the NIST announced the selection of a first set of Post-Quantum (PQ) algorithms to be standardized, addressing the question of integrating PQ cryptography in real-world protocols is urgent in order to anticipate difficulties and allow a smooth transition. This is especially true for banking applications where the ecosystem composed of a variety of cards and terminals is heterogeneous. Providing solutions to ensure efficiency and some kind of backward compatibility is mandatory. In this work, we provide the first analysis of card-based payments with respect to these questions. We integrate post-quantum algorithms in existing protocols, and propose hybrid versions. We implement them on banking smart-cards and analyse the impacts on various aspects of the product, from production to actual transactions with terminals. Our work shows that such products are possible, but we identify several issues to overcome in the near feature in order to keep the same level of usability.

Keywords: Post-quantum cryptography · Card-based payments · EMV · Hybrid protocols · Smart-cards.

1 Introduction

Though the path to a cryptographically relevant quantum computer is uncertain and may still be long, the time required to update our systems and the "record now, decrypt later" threat are requiring anticipation. Agencies and standardization bodies acknowledge this fact, and already launched actions towards a migration to Post-Quantum (PQ) cryptography. The NIST competition [24] is the main driver on this subject, and the announcement of the selected algorithms in July 2022 [25] is going to accelerate the development, standardization and adoption of post-quantum protocols.

In parallel, industry is invited to start considering this migration, by looking for potential weaknesses in systems and protocols, by anticipating the transitioning phase, and by experimenting with the technology as of today. It is important to identify as soon as possible the difficulties that will be faced, in particular in terms of performances. Indeed, post-quantum algorithms have generally much larger keys, ciphertext and signature sizes, and are more demanding in terms of computations and memory consumption. In addition to this "post-quantum

overhead", agencies are asking to first implement *hybrid protocols*, that is to say protocols that combine "classical", well studied cryptography, and PQ cryptography (see [4] for instance). This is to cope with the immaturity of the novel PQ algorithms, and prevent any regression. Once the community will have sufficient confidence, PQ algorithms will be used alone.

Some experiments with real world protocols have already been conducted. Back in 2016, Google deployed a hybrid scheme for key establishment in TLS [10]: the PQ algorithm NewHope [3,2] was used in addition to a classical key exchange. On the same subject, [29] formalizes the security of PQ versions of TLS that rely on key encapsulation mechanism instead of signature for authentication. We can also mention the proposal [30] for a hybrid version of the Signal protocol, also relying on PQ key encapsulation. More industry-oriented works include [27] which focuses on M2M protocols, and [9] on secure-boot in the automotive context.

In this work, we consider protocols used for card-based payments, according to EMV specifications [17]. Indeed, EMV transactions currently represent more than 90% of card-present payment transactions worldwide [3]. Those protocols make use of asymmetric cryptography (RSA) for card authentication by the terminal, and symmetric cryptography (usually TDES) for transaction certificates intended for the issuer. Contrary to protocols used to protect any kind of information, like TLS, the cryptographic data involved in a banking transaction are valuable only during a short period of time: after validation by the bank, they become useless. This makes banking transactions immune to the "record now, decrypt later" threat. On the other hand, *offline transactions* rely exclusively on card authentication. In this case, the validation by the bank is delayed: transactions certificates are stored some hours in the terminal before being transmitted to the bank. This is typically used to accelerate commercial exchanges in case of small amounts. Here, the authentication of the card is crucial for the security of the system. Though there is no short-term threat on RSA-based authentication, the industry should start today to explore PQ cryptography for this authentication step, so as to be ready for "Q-day". This is especially true because there are tight requirements on transactions timings, and more demanding computations could have a dramatic impact, as far as to require changes in the hardware of involved devices. This would be a very long process, that would add to the one of a cryptographic migration in the financial industry, itself long and complex. Early preparation and identification of obstacles can only facilitate this effort.

We here focus on the card part (including communications). As it is the most constrained device, this is where the repercussions are expected to be the greatest. To our knowledge, this is the first work that considers the integration of PQ cryptography into a banking protocol and that evaluates the corresponding impact on smart-cards in a real world context.

Contributions. We select two payment protocols: EMV CDA with offline PIN verification and offline transaction, and a variant providing protection against

[3] https://www.emvco.com/about/deployment-statistics/

user tracking. These protocols offer different security properties and work with different (classical) cryptographic algorithms. It is thus relevant to see how they behave with PQ algorithms. We propose PQ and hybrid versions of these protocols. For both of them, we add the PQ resistance by careful addition of cryptographic operations. We do this with several objectives in mind. First, and obviously, we maintain the security objectives of the original protocols. Second, we respect as much as possible the general structure in terms of commands and exchanges. This aims at easing the migration, by limiting modifications and allowing retro-compatibility. Finally, we attempt to reduce the overhead on transaction timings, and to this end we carefully design the PQ version of protocols. We select different PQ algorithms, all among the finalists of NIST's competition, and implement the result on smart-cards. We then analyze the impacts in terms of size of data (code size, card personalization and communications) and computations timings, and expose some areas of work that should be considered if we want to reach good performances with pure PQ or hybrid protocols.

Outline. Section 2 describes the existing classic protocols we consider. Section 3 explains how we integrated PQ cryptography in the selected protocols and presents the hybrid versions we implemented. Section 4 describes our implementations and analyses the results. Section 5 concludes this paper.

2 Existing Protocols

A vast majority of banking cards and terminals use the Europay, Mastercard and Visa (EMV) standard. This standard has been created in 1996, and covers data formats and protocols. For our study, we consider two protocols that we think are the most interesting. Here, we will recall the main steps of a transaction, briefly list the different available options, and then focus on the description of the cryptographic parts of the protocol we selected. For interested readers, [11] provides a more thorough overview of EMV.

The EMV specifications consist in several documents. We here recall the main phases of an EMV session, together with the security objectives. The interested reader can look at [17,18] for more detailed information. (Note that in practice, these phases can be interleaved into several commands.)

1. **Initialization.** During this phase, the reader selects the right application on card, and some data are transmitted by the card, including card number and expiry date, and supported features.
2. **Data authentication.** This stage is meant to ensure that the card is a genuine one. Three different methods are supported:
   - *Static Data Authentication (SDA).* In this case, the card provides signed data to the terminal. The latter can then check the authenticity of the data. Note that this method is prone to cloning.
   - *Dynamic Data Authentication (DDA).* Here, the card is equipped with an asymmetric key pair, the corresponding certificate, and the ability to perform signatures. A "challenge-response" exchange where the card

signs a challenge sent by the terminal allows to authenticate the card – in a way that prevents cloning.
There is however a security issue with DDA: though it allows authentication of the card, the terminal has no assurance that the same card is involved in subsequent exchanges.
- *Combined Data Authentication (CDA).* This method is similar to DDA, with the only difference that the card signature is also used to authenticate some data, so as to repair the flaw of DDA.

3. **Cardholder verification.** This can be done by PIN verification or handwritten signature. When PIN is used, this latter is entered on the terminal, and the verification happens either online – the PIN is sent to and checked by the issuer, or offline – the PIN is sent to the card. The PIN can be transmitted in clear text, or encrypted. In this case, the terminal encrypts the PIN for the card using its public key.
4. **Transaction.** This is the final step, it can be online, or offline:
   - *Online transaction.* In this case, the card provides an *Authorisation Request Cryptogram*, generated thanks to a symmetric key shared with the issuer. The terminal forwards it to the issuer, who verifies it. In case of approval, the card provides a *Transaction Certificate (TC)* as a proof of completion.
   - *Offline transaction.* Here, the card provides directly a TC, which is transmitted to the issuer later. The commercial transaction is authorized only on the basis of the card authentication.

A formal analysis of the protocol is provided by [28]. It confirms that only CDA properly answers the security objectives of card authentication, cardholder authentication and transaction authenticity, except for the attack [23]. Preventing it requires, in plus of using CDA, additional checks at the terminal level. These ones are out of the scope of our experiment, we will rather focus in preserving the features provided by CDA with PQ and hybrid security.

As can be seen, fully online transactions rely on symmetric cryptography, and are thus quite easily protected against quantum attacks: the already available 256-bit security symmetric cryptography provides quantum-resistance. However, those transactions cannot be executed everywhere (a good connection is required) and are less use-friendly, as they take more time to proceed. This is why offline transactions are often used. To protect these ones against quantum attacks, new PQ algorithms must be integrated. In the rest of this paper, we thus focus on offline transactions, with PIN transmitted encrypted and verified by the card, and delayed transmission of TC.

2.1 EMV CDA Protocol

For our analysis, we consider CDA, as it is the most secure variant. As explained above, we consider an offline transaction with PIN encryption. The resulting cryptographic protocol is depicted on Fig. 1. To ease reading and to focus on

the parts relevant to our purpose, we adapted EMV notations and omitted some details (including some hash computations that have no influence).

The protocol relies on the RSA algorithm, used both for encryption and signature. The card is equipped with a secret key $\mathsf{sk_C}$, and the corresponding certificate $\mathsf{cert_C}$, emitted by the issuer. The certificate $\mathsf{cert_{IS}}$ of the latter is also provided by the card. The first step is the terminal checking the chain up to the certification authority public key $\mathsf{pk_{CA}}$ (all certificates are signed with RSA). The function Verify is used to this aim: it takes as input a public key and a certificate, verifies the signature of the certificate and outputs the certified public key in case of success.

The function RSA.Enc (resp. RSA.Dec) takes as input a public RSA key (resp. a private RSA key) and a message to encrypt (resp. to decrypt), and outputs the ciphertext (resp. the plaintext). In a second step, the terminal uses it to encrypt the PIN entered by the user before sending it to the card. The card then performs the decryption and checks the PIN received against the reference value. Thanks to $nonce_C$, any replay of a previously recorded correct value is prevented.

The card is also equipped with a symmetric key $\mathsf{MK_{AC}}$, shared with the issuer and used to compute the cryptograms. These ones are computed thanks to a MAC function – usually based on DES or Triple-DES – applied on required information related to the transaction, denoted by $transInfo$. The card computes the cryptogram TC and signs it with its RSA secret key $\mathsf{sk_C}$, thanks to the function RSA.Sign. The value $nonce_T$ sent by the terminal is included in the signature. This allows the terminal to authenticate the card by verifying the signature with the card public key $\mathsf{pk_C}$, thanks to the function RSA.Verify. TC is stored in case of success.

2.2 BDH-based Protocol

In 2012, EMVCo decided to replace RSA by ECC-based cryptography, and expressed the need for a protocol establishing a secure channel between the card and the terminal, while preventing card tracking – this one is indeed possible as soon as the card transmits its certificate in plain. Two protocols were identified and drafted in [19]: one called *Blinded Diffie-Hellman (BDH)* – where the card uses a static Diffie-Hellman key to authenticate itself –, and a 1-sided version of the Station-To-Station protocol [15]. BDH is the most efficient one and has been proven secure by [12], and further discussed in [21].

This protocol substantially increases the general level of security. It is also more efficient than CDA, and does not make use of signatures. For these reasons, it is interesting to see how it can be made post-quantum, and to evaluate the cost of such a migration. Figure 2 sketches the main cryptographic steps of this protocol when it is used to complete an offline transaction with PIN encryption.

In this case, the card's secret key $\mathsf{sk_C}$ is an *Elliptic Curve Diffie-Hellman (ECDH)* private key, and the certificates of the trust chain ($\mathsf{cert_C}$, $\mathsf{cert_{IS}}$) are signed with EC-based signature. Given the elliptic curve $\mathcal{E}$ over the field $\mathbb{F}$, and the point G used as generator, the function EC.Blind takes as inputs an

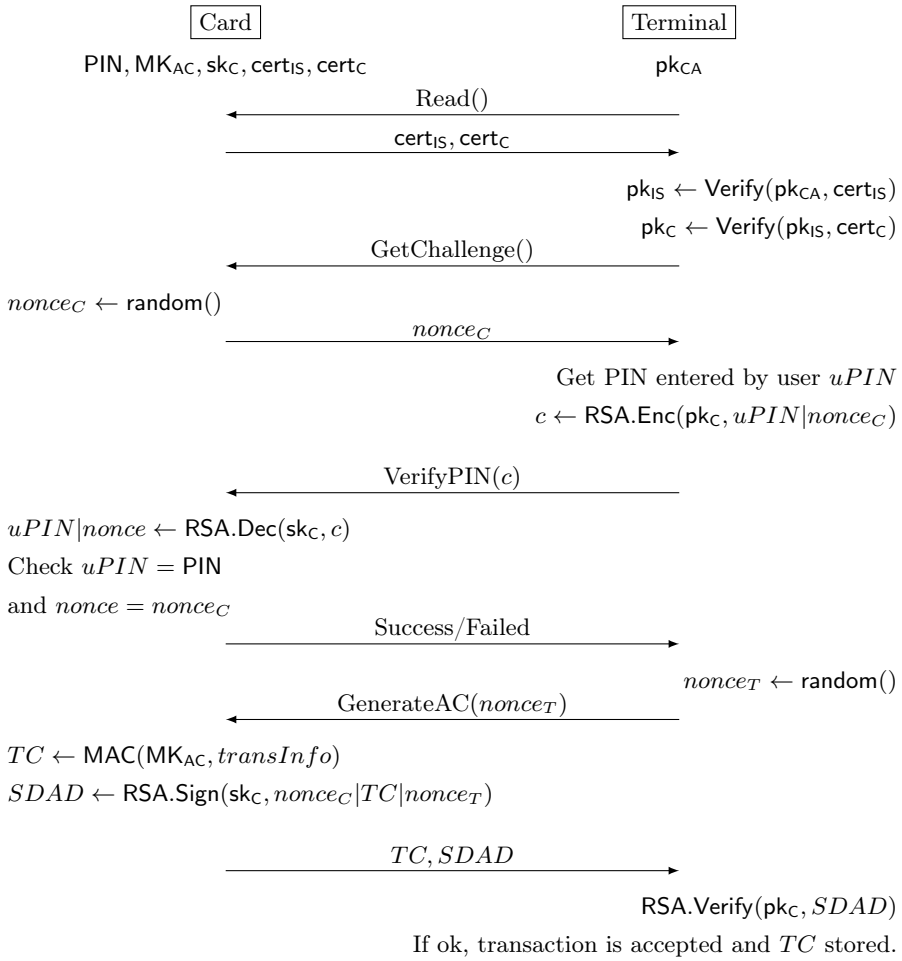

Fig. 1. EMV CDA protocol

element bf of $\mathbb{F}$ called the *blinding factor*, a private key sk also in $\mathbb{F}$, and the corresponding public point pk, with $pk = [sk] \cdot G$. It returns the blinded values $bsk = bf \cdot sk$ and $bpk = [bf] \cdot pk$. This function is used by the card to compute a *blinded* value of its static key pair (bsk_C, bpk_C). It then performs an ECDH agreement between (bsk_C and the just-received terminal ephemeral public key epk_T. A secure channel is established from the resulting shared secret shs by deriving a set of keys thanks to the function KDeriv. To ease reading, we denote K the keyset, even if it is composed of several keys. Functions AEnc and ADec are used for authenticated encryption and decryption, respectively. The card uses it to encrypt bf and send it to the terminal, together with the blinded public key bpk_C, allowing the terminal to establish K and recover bf.

Card's certificate chain is then sent encrypted to the terminal, which can verify the certificates, and authenticate the card by using the function EC.BlindVerif, which takes as input two points p_1 and p_2, a scalar s, and checks if $p_1 = [s] \cdot p_2$.

The secure channel is then used by the the terminal to securely communicate the PIN entered by the user to the card, and to verify the authenticity of the $SDAD$. It replaces the RSA operations used to this aim in the CDA-based transaction described in the previous section.

3 Post-Quantum and Hybrid Versions

In this section, we propose PQ versions of the transaction protocols described in Sect. 2. The new versions aim at achieving the same security properties as the original protocols, but using only PQ algorithms. After presenting the PQ variants, we address the problem of combining them with the classic protocols to achieve hybrid versions.

The NIST standardization process [24] covers two types of cryptographic primitives: digital signature and key encapsulation mechanism.

Definition 1 (Key Encapsulation Mechanism). *A key encapsulation mechanism (KEM) is an asymmetric cryptographic primitive that allows two parties to establish a shared secret key. Given a private/public key pair* (sk, pk) *provided by a* KeyGen *primitive, the* Encaps *primitive takes the public key* pk *as input and outputs a pair* (ss, ct) *composed of a random secret value and a ciphertext. The* Decaps *primitive takes as input the private key and the ciphertext and recovers the secret value.*

$$\begin{aligned} (\mathsf{sk}, \mathsf{pk}) &\leftarrow \mathsf{KeyGen}() \\ (ss, ct) &\leftarrow \mathsf{Encaps}(\mathsf{pk}) \\ ss &\leftarrow \mathsf{Decaps}(\mathsf{sk}, ct)\,. \end{aligned}$$

Compared to KEM, performing a signature is more expensive in general, both in speed and in memory consumption. This is especially important when implemented in embedded devices [22]. For these reasons, our strategy is to

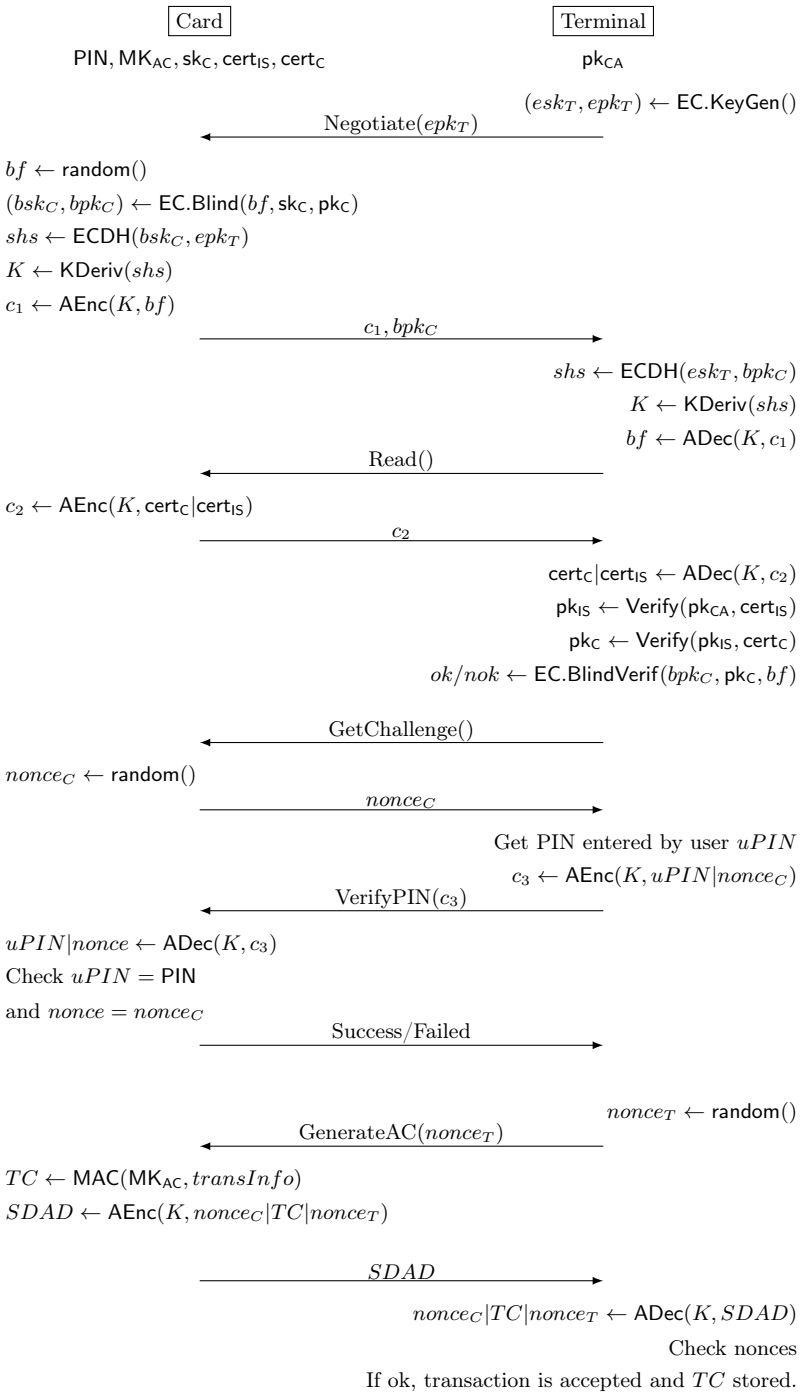

Fig. 2. BDH-based protocol

only use signature when an actual signature is required (e.g. certificates). For operations whose goal is to authenticate, we achieve the same function using only key encapsulation mechanisms. To differentiate between classic crypto keys and PQ crypto keys, we use a star to denote a PQ key (e.g. pk_C^*).

For parts using symmetric cryptography, the algorithms can simply be replaced by 256-bit key AES, which is enough to achieve a 128-bit security against quantum computers, without modifying the transaction flow. Thus, we will omit this topic in the following, and focus only on asymmetric cryptography.

3.1 PQ Version of EMV CDA Protocol

In Fig. 3, we present a post-quantum analog of the CDA protocol described in Fig. 1. In this version, the terminal challenges the card by asking to decapsulate a ciphertext generated for the card's public key. This step replaces the signature performed in Fig. 1. Not only the card can be authenticated with respect to its public key, but the shared secret can be used to switch to symmetric cryptography for the rest of the exchanges instead of using public key encryption for PIN transmission and signature for card and TC authentication, as in Fig. 1. In order for the card to introduce its own randomness, the $nonce_C$ generated in the GetChallenge is incorporated by both parties in the derivation of the shared key K. This very value K can then be used to encrypt and authenticate the user-entered PIN denoted by $uPIN$ for the card while still providing anti-replay protection. It is also used to authenticate the card and TC towards the terminal. Note that the sequence adds the EstablishKey command after the GetChallenge for the terminal to send its ciphertext to the card. It is worth noticing that one could merge both commands GetChallenge and EstablishKey to reduce the number of exchanges.

3.2 PQ Version of BDH-based Protocol

The BDH-based protocol uses the mathematical properties of elliptic curves to derive a valid ephemeral public key from a static public key using the blinding factor. It is not possible to achieve this in general with PQ algorithms. To achieve authentication, we can use a protocol inspired by KEM-TLS [29]. In Fig. 4, the terminal generates an ephemeral key pair, and sends the public part to the card. The card can then encapsulate a secret and derive a first shared key set K_1 to securely communicate with the terminal. This is performed thanks to the NegotiateKEM command which replaces the Negotiate one in the classic version. The shared key set K_1 is used by the card to send its static public key encrypted, so that it cannot be linked to the actual user by an eavesdropper.

Next, the terminal can authenticate the card's public key, and challenge the card by asking the decapsulation of the ciphertext ct_2. To completely ensure privacy, this value is sent encrypted with K_1. This is to avoid the case where the terminal's encapsulation is fully deterministic (e.g. bad random, or random derived from pk_C^*). Two passes are necessary to achieve the same privacy-preserving properties of the classic BDH-based protocol. Once the new derived secret K is

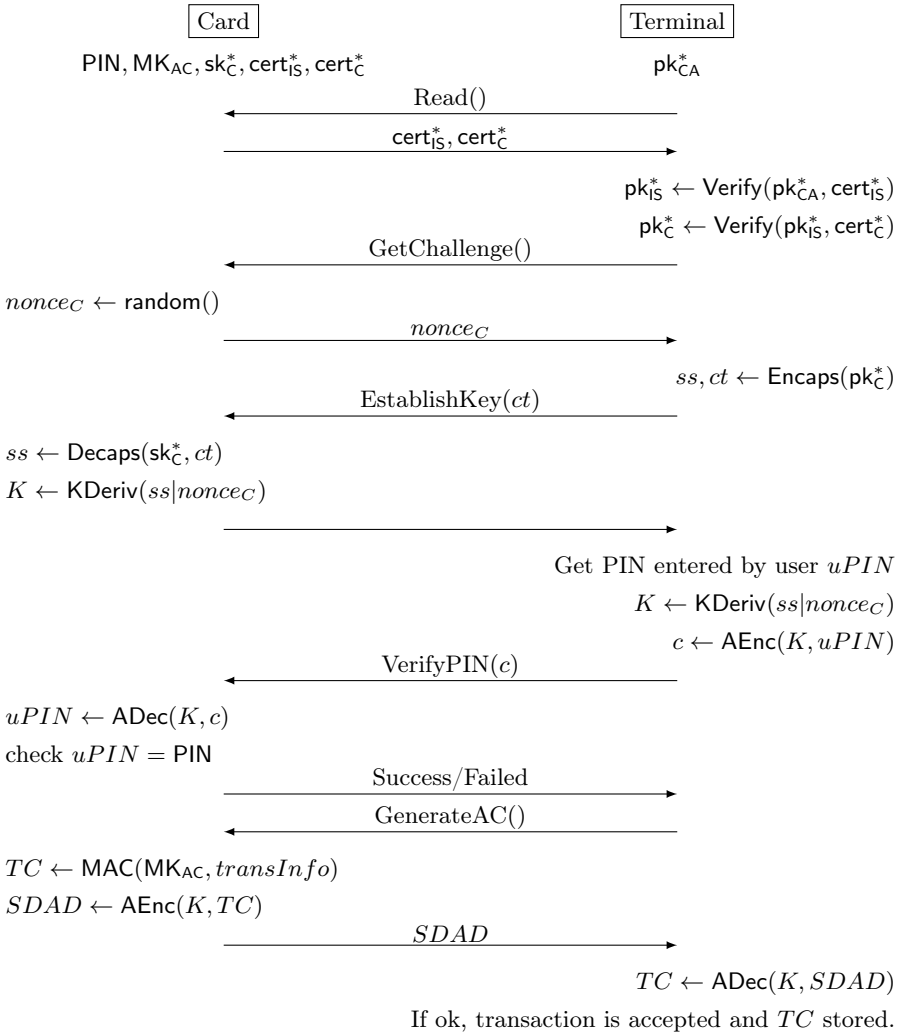

Fig. 3. PQ analog of CDA protocol

shared between the card and the terminal, the PIN encryption and the rest of the transaction can be conducted in the same way as in Fig. 2, so this part is omitted in our description. The derivation from both ss_1 and ss_2 ensures anti-replay of the encrypted PIN and card/TC authentication at the same time.

Finally, please note that this protocol offers implicit authentication by sharing the common secret K. This can be made explicit by sending an acknowledgment encrypted with the shared key K to the terminal in response to the EstablishKey command if required.

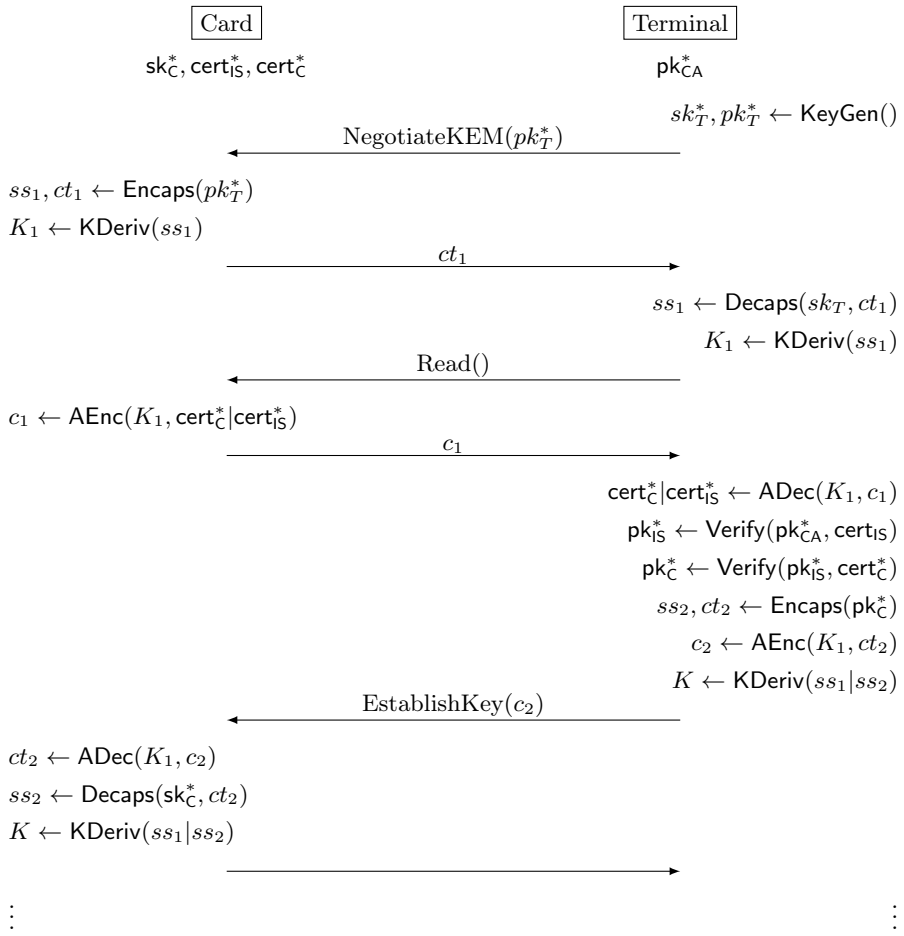

Fig. 4. PQ analog of BDH-based protocol

3.3 Hybrid Versions

Though our PQ protocols do not make use of signatures, these ones are needed for certificates. Indeed, two certificates are sent by the card and verified by the terminal: the one of the issuer, and the one of the card. For hybrid versions, different approaches exist, one having two separated chains, and others using "hybrid certificates" (see [8] for instance). All options show no significant difference in terms of size: in all cases, the same number of public keys and signatures are present. As a consequence, we decided to implement separated chains for simplicity. Of course, considering certificate management and performances on the terminal side could lead to a different implementation.

For the CDA protocol, we can derive a hybrid parallel by arranging together protocols presented by Fig. 1 and Fig. 3 as follows:

- The card stores both RSA and PQ KEM key pairs and certificates.
- In the first Read command, the certificates chain of the PQ static public key is sent to the terminal. The terminal then verifies the certificate chain.
- The terminal then asks the card to decapsulate ct, the encrypted shared secret ss.
- In a second Read command, the certificates chain of the classic static public key is sent to the terminal. The terminal then verifies the certificate chain.
- After the terminal receives $nonce_C$, a key K is derived from $nonce_C$ and the PQ shared secret ss. The $uPIN$ entered by the user is first encrypted using K (PQ security), then using RSA (classic security):
  $c \leftarrow \mathsf{RSA.Enc}(\mathsf{pk_C}, \mathsf{AEnc}(K, uPIN))$.
- The TC is authenticated using the key K, and is also signed using the RSA private key as in Fig. 1: $\mathsf{RSA.Sign}(\mathsf{sk_C}, nonce_C|TC|nonce_T)$. Both values are sent as a response to the GenerateAC command.

The result is described in Fig. 5.

For the BDH-based protocol, we derive a hybrid version by arranging protocols presented by Fig. 2 and Fig. 4 as follows:

- The card stores both RSA and PQ KEM key pairs and certificates.
- Both Negotiate and NegotiateKEM – unchanged with respect to classic and PQ versions respectively – are used, so that the card can send classic certificates encrypted with an ECDH-derived key K_0, and PQ certificates encrypted with a KEM-derived key K_1.
- The terminal can check all certificates, and derive the secret K not only from the PQ shared secrets ss_1 and ss_2 (as in the PQ version), but also from the shared secret shs obtained from classic BDH: $K \leftarrow \mathsf{KDeriv}(shs|ss_1|ss_2)$.

The result is described in Fig. 6.

4 Practical Implementation

In this section, we describe our implementations and give the results of our measurements. We analyse them so as to identify points that require specific attention in view of a post-quantum migration.

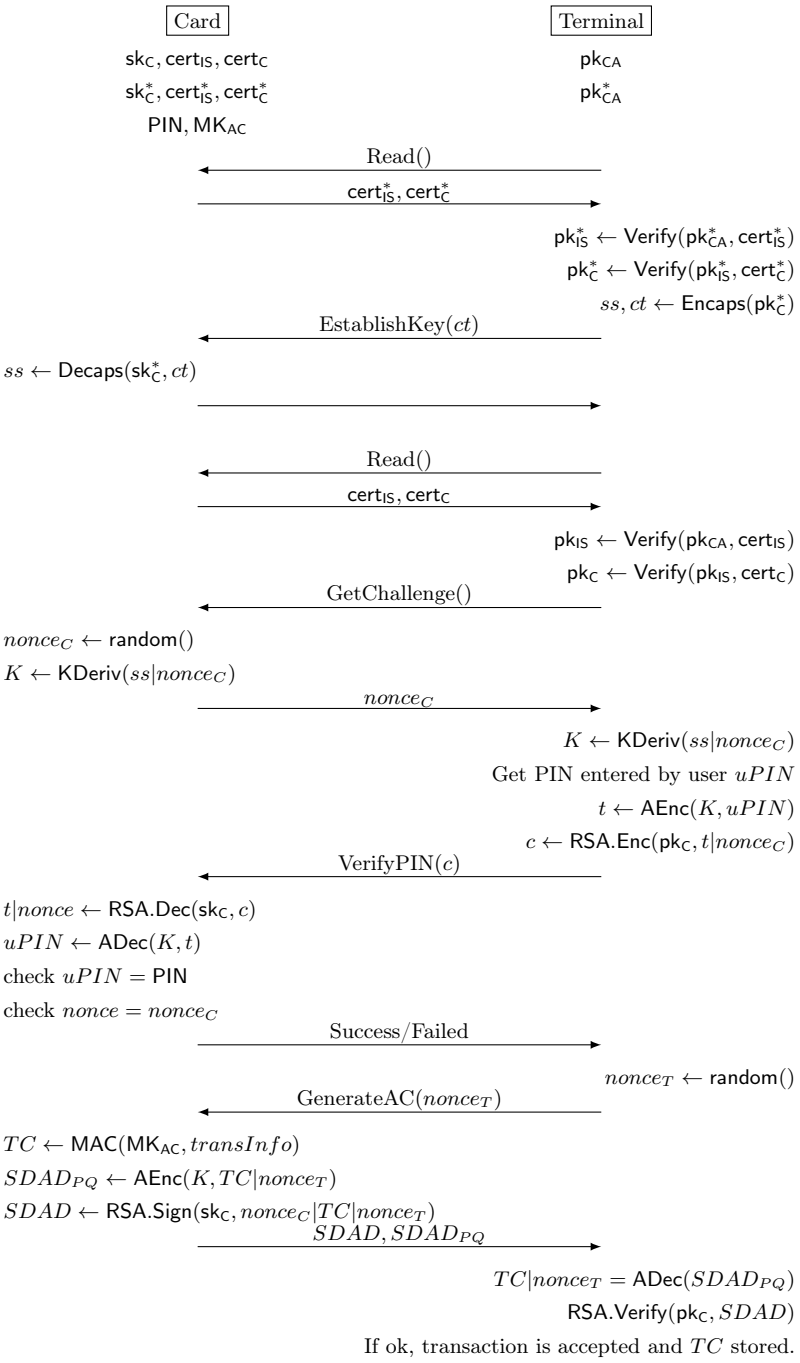

Fig. 5. Hybrid analog of CDA protocol

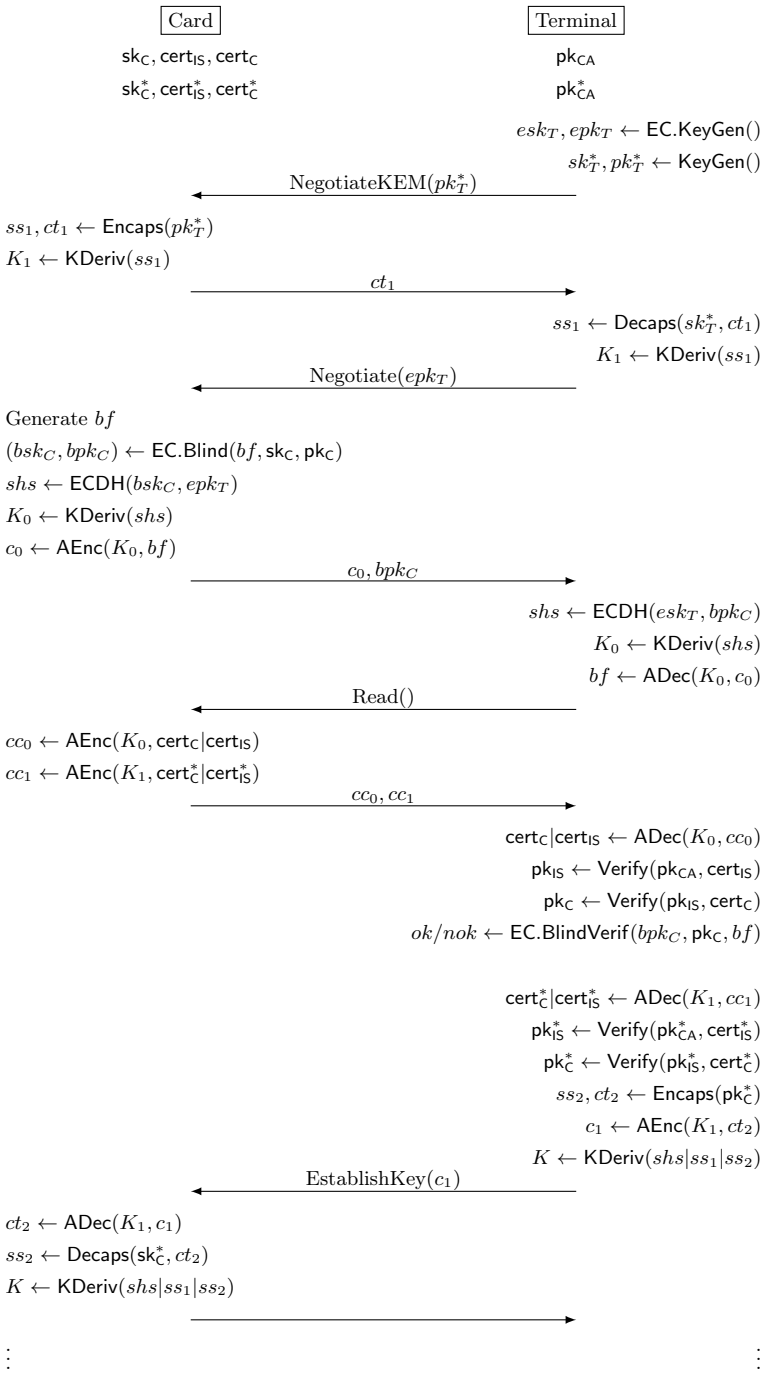

Fig. 6. Hybrid analog of the BDH-based protocol

4.1 Post-Quantum Algorithms Selection

In order for this work to be relevant for further experiments, we based our selection on NIST's positions. Indeed, we can expect that most standardization bodies will articulate their works towards PQ protocols based on NIST standards.

Some hash-based signature algorithms are already standardized by NIST [14]. They enjoy high confidence from the community and thus can be used alone (in a non-hybrid way). However, unlike general purpose signatures schemes, they are state-full, and as such, they require specific management. Despite the gain in size they could bring in our case, we decide not to use them, as we feel the gain is not large enough to justify putting more constraints on the back-end. It is of course debatable, and advances in the management of these signatures schemes (in particular in Hardware Security Modules) could justify their use. However, as said, the gain is limited and would not change drastically our figures. Thus, we considered the two signature algorithms that have been finally selected for standardization: Falcon [20] and CRYSTALS-Dilithium [6]. We first implemented candidates of security strength category 1, according to NIST's definition (categories range from 1 to 5, cat. 1 being the (somehow) equivalent of AES with 128-bit key, and cat. 5 the equivalent of AES with 256-bit key). However, as will be shown in our figures, we quickly realised that the size of certificates was the most impacting characteristic regarding transaction timings. We thus made some more experiments with signatures in categories 3 and 5, in order to measure the behaviour of timings with respect to certificates security level. Indeed, it seems reasonable to think that different security levels would be required for the different actors in the certificate chain, depending on their lifetime.

Our PQ and hybrid versions of CDA and BDH make use of a KEM. Several characteristics of the chosen algorithm are relevant for our experiment:

- the card needs to store the KEM secret key;
- the KEM public key is transmitted as part of card's certificate;
- one (for PQ CDA) or two (for PQ BDH) KEM ciphertexts are exchanged;
- the card will perform one Decaps operation, and also one Encaps in case of PQ BDH.

When we started this work, four KEM algorithms were still in competition in the final round of NIST's competition: Classic McEliece [1], CRYSTALS-Kyber [5], NTRU [13] and SABER [7]. McEliece has key size incompatible with embedded usage. The three remaining have roughly similar public key and ciphertext sizes, but NTRU has shorter secret key. We thus selected NTRU. In the meantime, Kyber has been chosen for standardization, so we also made the experiments with it as a KEM. As NTRU is the "fall-back" solution in case no agreement is found with some patent owners (sees report [26]), we think it is interesting to present the results in this case also.

More specifically, for NTRU we implemented the variant ntruhps2048509, and for Kyber kyber512. Both are category 1 algorithms. As smart-cards have a limited lifetime, and limited capabilities, this security level seems to offer the best trade-off.

4.2 Implementation Description

The protocol has been implemented on an ARM Cortex-M3 derivative, which is a common architecture for banking smart-cards. Several applications support EMV protocols, we opted for a *Common Payment Application (CPA)*, compliant with [16], in which we add support of the hybrid protocols presented earlier. We use a basic, "toy" profile for the application. The version with the EMV-compliant CDA protocol described by Fig. 1 serves as a reference for timings and memory size. In practice however, very different figures can be observed depending on many parameters, including the chip and the profile of the target application.

The protocols presented in Sect. 2 are implemented through commands defined by EMV. We reused these commands as much as possible for the hybrid versions, however, some modifications have been necessary.

We modified two commands involved in the first phase, where the terminal selects the payment application and gets the data necessary to authenticate the card. First, the `SELECT_APPLICATION` now allows the card to indicate whether it supports a hybrid version of the protocol, and which signature is used in the certificates. Second, as EMV does not provide any command to read data of length superior to 256 bytes, we developed a new command `GET_DATA_EXTENDED` to this end. It is used to read $\mathsf{cert}_{\mathsf{IS}}$ and $\mathsf{cert}_{\mathsf{C}}$.

In addition, the hybrid versions require some operations related to the KEM scheme, which are not supported by EMV. We implemented a dedicated command `PERFORM_PQ_OPERATION` and use it to perform the Encaps and Decaps operations, whenever required.

Finally, the original EMV command for PIN verification has been updated to decrypt the PIN using the key K. The command for transaction certificate generation has also been updated to generate $SDAD_{PQ}$ in addition to $SDAD$, as described in Fig. 5.

4.3 Performances Analysis

Memory size. Regarding code, the implementation of the KEM Encaps and Decaps operations, of the two additional commands and other modifications described above amounts to 5.51 KB for ntruhps2048509 and to 3.47 KB for kyber512. Chips used for this kind of applications usually have several hundreds of kilobytes of Flash memory, so this overhead is manageable in both cases.

Regarding user-dependent data, which has to be loaded on the card during a specific operation called *personalization*, the PQ part of the protocols adds:

- $\mathsf{cert}^*_{\mathsf{IS}}$, which mainly includes the issuer's signature public key, and the signature by the authority,
- $\mathsf{cert}^*_{\mathsf{C}}$, which mainly includes the card's KEM public key, and the signature by the issuer,
- $\mathsf{sk}^*_{\mathsf{C}}$, the card's KEM secret key.

We consider the size of an instance with or without the PQ layer, and deduce the overhead. We perform the exercise for all proposed versions of Falcon and

Dilithium: Falcon-512 and Falcon-1024 (resp. cat. 1 and 5) and cat. 2, 3 and 5 versions of Dilithium, denoted in the rest of the document by Dilithium-2, Dilithium-3 and Dilithium-5 respectively. Results are shown in Tab. 1 for ntruhps2048509, and in Tab. 2 for kyber512. In both cases, we first give the *theoretical* overhead, i.e. the one obtained by considering only the size of involved data. We then give the overhead as measured in practice. We notice a difference between the two columns. This is due of course to some headers but most importantly, to the fact that writing in Flash memory is done *per page*, and that the considered chip's pages are 1024-byte long. This is why the actual numbers are multiples of 1024.

The increasing factor demonstrates how significant the impact of PQ cryptography is. In our case, it is mainly due to the size of certificates. Both KEM give similar increase, with Kyber being slightly worse due to a larger public key. For Dilithium-5, we reached the maximum capacity of our application and were unable to load the data. This shows that in certain cases, this migration will require modifications to allow more space for user data, and so adaptation of the hardware or optimizations to free some space. This figure has also an impact on the production. Indeed, as we said, these user-dependent data are introduced during the personalization, and any extension of the timing of this operation lowers the production capacity.

Table 1. Personalization overhead (NTRU)

Algorithm	Cat.	Overhead (bytes)		Increase (%)
		Theoretical	Measured	
Falcon-512	1	3863	4096	34.29
Falcon-1024	5	5987	6144	43.91
Dilithium-2	2	7786	8192	51.07
Dilithium-3	3	10 172	10 240	56.61
Dilithium-5	5	13 416	N/A	N/A

Table 2. Personalization overhead (Kyber)

Algorithm	Cat.	Overhead (bytes)		Increase (%)
		Theoretical	Measured	
Falcon-512	1	4661	5120	39.48
Falcon-1024	5	6785	7168	47.73
Dilithium-2	2	8584	9216	54.01
Dilithium-3	3	10 970	11 264	58.93
Dilithium-5	5	14 214	N/A	N/A

Transaction timings. Tables 3 and 4 present the timings of transactions for hybrid versions of the CDA and BDH-based protocols, for the different signature algorithms we used in certificates, with NTRU and Kyber as KEM respectively. To ease the analysis, we measure separately the communication timings, for which we distinguish terminal-initiated (`T->C`) and card-initiated (`C->T`) transmissions, and the card processing timing (terminal processing time is negligible comparatively). The total timing is indicated, as well as the ratio to a classic CDA transaction.

Overall, the global timing of transactions stay in conceivable orders of magnitude. They might however be too high to be used in practice. The ratio column exhibits indeed the huge impact hybridisation has on the transaction timing. We can observe that most of the transaction timing is due to the transmission of the card certificate chain to the terminal. This is due to the large size of PQ certificates, and the relative slowness of the card-terminal interface. This latter has not been designed to support such exchanges and thus severely slows down the transaction.

The processing on the card is relatively reasonable, and only increases a little when the certificates get larger — this is due to the processing of more commands to send them. When comparing NTRU and Kyber, we note that Kyber is faster, but that this advantage is lost in the transmission of a larger public key in card's certificate. At the end, both options gives very similar figures.

We should however keep in mind that our implementations of ntruhps2048509 and kyber512 are not secure against side-channel attacks. Implementing countermeasures against differential power analysis and fault attacks could bring significant overhead, up to doubling the execution time. Yet this would not change the conclusion: the communication part would still be overwhelming. Furthermore, future optimizations of algorithms and implementations, and dedicated hardware accelerators should improve the performances of the embedded part. We can also note that the hybrid BDH-based protocol is slower than the hybrid CDA one. This is mainly due to the additional Encaps operation required to provide tracking resistance.

5 Conclusion

By carefully designing hybrid protocols that take into account the specificities of the current algorithms involved in the PQ standardization, we propose a solution that could be considered for future specifications of banking transaction protocols. In particular, we took care to make use of digital signatures only when mandatory. We demonstrated the relevance and the feasibility of our propositions by implementing them on a real device. Our work allows to estimate the overhead of an hybrid version compared to the current protocols. It appears that a large part of the processing time is due to the transmission of certificates. In this context, a PQ signature scheme with small signatures would be the best choice. Further work is needed to secure the embedded implementation against side-channel attacks, and to provide proofs of security for the protocols.

Table 3. Full transaction timings for hybrid protocols (NTRU)

Protocol	Certificate Algo.	Timings (ms)				Ratio
		Comm.		Card proc.	Total	
		T->C	C->T			
Hybrid CDA						
	Falcon-512	1075	4039	390	5504	3.28
	Falcon-1024	1111	6030	390	7531	4.49
	Dilithium-2	1155	7710	398	9263	5.53
	Dilithium-3	1197	9947	416	11 560	6.90
Hybrid BDH						
	Falcon-512	1772	4753	450	6975	4.16
	Falcon-1024	1798	6746	479	9023	5.38
	Dilithium-2	1826	8436	501	10 763	6.42
	Dilithium-3	1867	10 696	530	13 093	7.81

Table 4. Full transaction timings for hybrid protocols (Kyber)

Protocol	Certificate Algo.	Timings (ms)				Ratio
		Comm.		Card proc.	Total	
		T->C	C->T			
Hybrid CDA						
	Falcon-512	1134	4133	376	5643	3.36
	Falcon-1024	1170	6132	376	7678	4.58
	Dilithium-2	1214	7804	384	9402	5.61
	Dilithium-3	1256	10 041	402	11699	6.98
Hybrid BDH						
	Falcon-512	1917	4908	354	7179	4.28
	Falcon-1024	1967	6908	383	9258	5.52
	Dilithium-2	1971	8511	405	10887	6.50
	Dilithium-3	2012	10 851	434	13297	7.93

References

1. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic McEliece: conservative code-based cryptography. Tech. rep. (2020), https://classic.mceliece.org/
2. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: NewHope without reconciliation. Cryptology ePrint Archive, Report 2016/1157 (2016), https://eprint.iacr.org/2016/1157
3. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 327–343. USENIX Association (Aug 2016)
4. ANSSI: ANSSI views on the Post-Quantum Cryptography transition (2022), https://www.ssi.gouv.fr/en/publication/anssi-views-on-the-post-quantum-cryptography-transition/
5. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber – Algorithm Specifications and Supporting Documentation. Tech. rep. (2021), https://pq-crystals.org/kyber/index.shtml, version 3.2
6. Bai, S., Ducas, L., Kiltz, P., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium – Algorithm Specifications and Supporting Documentation. Tech. rep. (2021), https://pq-crystals.org/dilithium/, version 3.1
7. Basso, A., Mera, J.M.B., D'Anvers, J.P., Karmakar, A., Roy, S.S., Beirendonck, M.V., Vercauteren, F.: SABER: Mod-LWR based KEM (Round 3 Submission). Tech. rep., https://www.esat.kuleuven.be/cosic/pqcrypto/saber/
8. Bindel, N., Herath, U., McKague, M., Stebila, D.: Transitioning to a quantum-resistant public key infrastructure. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017. pp. 384–405. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-59879-6_22
9. Bos, J.W., Carlson, B., Renes, J., Rotaru, M., Sprenkels, D., Waters, G.P.: Post-quantum secure boot on vehicle network processors. Cryptology ePrint Archive, Paper 2022/635 (2022), https://eprint.iacr.org/2022/635
10. Braithwaite, M.: (2016), https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html
11. van den Breekel, J., Ortiz-Yepes, D.A., Poll, E., de Ruiter, J.: EMV in a nutshell (2016), https://www.cs.ru.nl/~erikpoll/papers/EMVtechreport.pdf
12. Brzuska, C., Smart, N.P., Warinschi, B., Watson, G.J.: An analysis of the EMV channel establishment protocol. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 373–386. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516748
13. Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z.: NTRU – Algorithm Specifications and Supporting Documentation. Tech. rep. (2020), https://ntru.org/
14. Cooper, D.A., Apon, D.C., Dang, Q.H., Miller, M.S.D.M.J.D.C.A.: Recommendation for stateful hash-based signature schemes. Tech. rep., NIST (2020), https://doi.org/10.6028/NIST.SP.800-208
15. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes and Cryptography **2**, 107–125 (1992)

16. EMVCo: EMV – Integrated Circuit Card Specifications for Payment Systems – Common Payment Application Specification (2005), version 1.0
17. EMVCo: EMV – Integrated Circuit Card Specifications for Payment Systems – Book 2 – Security and Key Management (2011), version 4.3
18. EMVCo: EMV – Integrated Circuit Card Specifications for Payment Systems – Book 3 – Application Specification (2011), version 4.3
19. EMVCo: EMV ECC Key Establishment Protocols (2012)
20. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Tech. rep. (2020), https://falcon-sign.info/
21. Garrett, D., Ward, M.: Blinded Diffie-Hellman. In: Chen, L., Mitchell, C. (eds.) Security Standardisation Research. pp. 79–92. Springer International Publishing, Cham (2014)
22. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: Testing and benchmarking NIST PQC on ARM cortex-m4. IACR Cryptol. ePrint Arch. p. 844 (2019), https://eprint.iacr.org/2019/844
23. Murdoch, S.J., Drimer, S., Anderson, R., Bond, M.: chip and pin is broken. In: 2010 IEEE Symposium on Security and Privacy
24. National Institute for Standards and Technology: Post-Quantum Cryptography Standardization, https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization
25. National Institute for Standards and Technology: Selected Algorithms 2022, https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms-2022
26. National Institute for Standards and Technology: Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process (July 2022), https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf
27. Paul, S., Scheible, P.: Towards post-quantum security for cyber-physical systems: Integrating PQC into industrial M2M communication. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12309, pp. 295–316. Springer (2020)
28. de Ruiter, J., Poll, E.: Formal Analysis of the EMV Protocol Suite. In: Mödersheim, S., Palamidessi, C. (eds.) Theory of Security and Applications. pp. 113–129. Springer Berlin Heidelberg (2012)
29. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 20. pp. 1461–1480. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3423350
30. Stadler, S., Sakaguti, V., Kaur, H., Fehlhaber, A.L.: Hybrid signal protocol for post-quantum email encryption. Cryptology ePrint Archive, Paper 2021/875 (2021), https://eprint.iacr.org/2021/875, https://eprint.iacr.org/2021/875