Preparing Passports for the Post Quantum Era

Securing Travel Documents with Post Quantum Cryptography
Master thesis

by

Siebren Lepstra

Student number:	852064786	
Course code:	IM9906	
Thesis committee:	dr. Greg Alpár,	Radboud University, Open University
	dr. ir. Hugo Jonker,	Open University

Contents

Acronyms

AA Active Authentication. 7, 21

APDU Application Protocol Data Unit. 45

BAC Basic Access Control. 7

BSI Bundesamt für Sicherheit in der Informationstechnik. 24

CA Chip Authentication. 7, 16

CAN Card Access Number. 16

CMS Cryptographic Message Syntax. 45

CSCA Country Signing Certificate Authority. 7, 19

CVC Card Verifiable Certificates. 25

CVCA Country Verifying Certificate Authority. 25

DH Diffie-Hellman. 15, 16

DS Document Signer. 7, 19

DVCA Document Verifying Certificate Authority. 25

EAC Extended Access Control. 7, 25, 52

ECDH Elliptic Curve Diffie-Hellman. 15, 16

EEPROM Electrically Erasable Programmable Read-only Memory. 11

EF.SOD Document Security Object. 19, 45

eMRTD Electronic Machine Readable Travel Document. 6, 8, 10, 11, 19

ENISA European Union Agency for Cybersecurity. 11, 32

ICAO International Civil Aviation Organisation. 6

IS Inspection System. 25

JMH Java Microbenchmarking Harness. 36

KEM Key Encapsulation. 53

KEX Key Exchange. 53

LDS Logical Data Structure. 6, 19, 45

MRZ Machine Readable Zone. 13, 16

NIST National Institute of Standards and Technology. 9

NPKD National Public Key Directory. 19

OTS One-Time Signature Scheme. 9, 31

PA Passive Authentication. 7, 9, 22

PACE Password Authenticated Connection Establishment. 7

PKI Public Key Infrastructure. 7, 19, 25

SSC Send Sequence Counter. 13

TA Terminal Authentication. 7, 25

ABSTRACT

Quantum computers pose a significant threat to electronic machine readable travel documents, such as passports, which are used for long periods, from 10 to 15 years. This long lifespan of these documents makes the potential impact of quantum computers even more critical, as the current security protocols rely on classical asymmetric cryptography which will be vulnerable in a post-quantum era. In particular, passive authentication, the protocol responsible for safeguarding the integrity of the travel document, is at risk. This research evaluates and benchmarks three quantum-resistant signature algorithms currently being standardized by NIST: Dilithium, Falcon, and SPHINCS+. The findings indicate that Dilithium and Falcon are well-suited to replace the Public Key Infrastructure (PKI) used in passive authentication due to their small digital footprint. SPHINCS+ is, by contrast, less suitable for this application due to its large signature size. Consequently, this study establishes a new quantum-resistant PKI and eMRTD, marking a significant step forward in securing travel documents against future quantum threats.

1. Introduction

Quantum computing promises to make traditional approaches to cryptography obsolete. Asymmetric cryptography is based on mathematical problems that are hard to solve for traditional computers. Not for quantum computers: by running Shor's algorithm [Sho97], they can efficiently break these problems. This has led to a significant research effort into so-called post-quantum cryptography, that is, cryptographic building blocks that are resistant to quantum computing. Designing post-quantum cryptography is only the first step though. Cryptography is widely deployed and in every-day use, both in software (e.g., browsers, messaging apps, and video calling apps) and in hardware (e.g., access tokens such as access cards, passports, and public transport cards). While changing the cryptographic primitives of software is relatively straightforward (i.e., update the software), this is not true for hardware. For example, it is not clear whether the hardware design of the tokens or of their readers needs to be updated to accommodate post-quantum cryptography. Answering such questions naturally depends on which post-quantum algorithms are to be used. This research proposal aims to investigate these questions for passports, that is, this proposal aims to find out how to make passports post-quantum proof.

Passports are not exclusive to the modern era, the first known written permission to travel was discovered in the Hebrew Bible (Nehemiah 2:7–9, around 450 BC) [DD92]. In medieval times, safe conducts were provided by monarchs to provide safe passage through the kingdom [McH21]. Although the safe-conducts have similarities with modern passports, it did not define an individual's citizenship at the time [McH21]. The modern variant of the passport began taking its shape starting in 1968 where the International Civil Aviation Organisation (ICAO) started work on Electronic Machine Readable Travel Document (eMRTD)s and the first edition of the ICAO Doc. 9303 was released in 1980 [ICA21a]. The ICAO 9303 is the standardization document for electronic travel documents. In the late eighties, discussion began about integrating a chip in the passport to improve the level of security of travel documents [DD92]. The standardization of the chip structure (LDS) and the required infrastructure for electronic travel documents was introduced into the sixth edition of the ICAO 9303 in 2006 [ICA21a]. Also today, passports are an important part of securing the national borders.

Passports are secured physically and, if they contain a chip, also digitally. When the passport is equipped with a chip, it is called an Electronic Machine Readable Travel Document (eMRTD). The International Civil Aviation Organisation (ICAO) has standardized [ICA21b, ICA21c, ICA21d] the protocols, mechanisms and infrastructures which are required in order to establish secure access to the chip, achieve data integrity and authenticity and to prevent unauthorized access to privacy-sensitive elements of the passport. The data of a passport is secured on the chip in such a way, that physical access is required to read the chip data of a travel document. In order to read a travel document, Basic Access Control (BAC) or Password Authenticated Connection Establishment (PACE) must be performed. Since 2018, PACE-only access has become mandatory for documents issued [ICA21c]. After performing BAC or PACE, electronic access is granted to the (electronic counterparts of the) less privacy-sensitive visible parts of the eMRTD, which corresponds with the visible data on the holder page of the travel document. Optionally, privacy-sensitive information such as fingerprints and other biometric features may be stored in the chip [ICA21c]. For documents in the EU, this is a mandatory requirement for fingerprints. The privacy-sensitive parts of the document, can only be accessed by performing Extended Access Control (EAC). EAC performs Chip Authentication (CA) and Terminal Authentication (TA) consecutively. CA establishes a secure channel between the chip and the reader using a Diffie-Hellman key agreement and TA determines if the terminal reading the document is genuine and allowed to read the contents of the privacy-sensitive data on the chip [RS14]. The chip is also secured against unauthorized modifications by Passive Authentication (PA) [ICA21c]. Examples of modifications are altering personal details of the passport holder to avoid detection or changing security information such as the public key stored in data groups 14 and 15 used for AA and CA. PA is the verification process of verifying the signed data of the passport by using a Public Key Infrastructure (PKI) and its chain of trust in order to determine if a document is genuine. When a passport is produced, the hashes of the data on the passport is signed by a Document Signer (DS) and the signature and the document signer are stored on the chip. When performing a check at the border, the document signer is checked to verify that it is signed by a trusted Country Signing Certificate Authority (CSCA). Subsequently, the signature of the data groups is checked in order to verify if the passport is not modified. This is the most important process of document verification, because this process distinguishes forgeries from genuine passports. Another security measure is Active Authentication (AA) to counter cloning of the chip [ICA21c]. This process performs a challenge-response protocol based on the private key located in a secure location of the chip. This secure location can only be accessed by the chip itself, so when a cloned chip performs AA, the process will fail because the cloned chip does not have the original key of the cloned passport. When using a wrong private key during active authentication, the only conclusion for the reader is that the passport has been cloned.

Each mentioned protocol and implementation uses symmetric and asymmetric cryptosystems [ICA21c]. The imminent arrival of quantum computing is a threat for these cryptosystems. The used asymmetric cryptosystems are vulnerable due to Shor's quantum algorithms for computing prime factorisation and discrete logarithms [Sho97]; symmetric cryptosystems are vulnerable due to Grover's algorithm, which speeds up key recovery attacks [Gro96]. The key strength in bits of symmetric algorithms are cut in half in a post-quantum era [RMYK17]. This would mean for example that the key strength of AES256 is reduced to 128 bits. Another cryptographic method that is used alongside the eMRTD protocols is hashing [ICA21c]. Similar to the symmetric cryptosystems, hashing algorithms are vulnerable to Grover's algorithm combined with the birthday paradox for collision attacks [MVZJ18, BHT98]. However, such attacks are expensive to perform on a quantum computer and the most efficient quantum algorithm for finding collisions reduces the search domain of n-bit hash algorithms to $2^{n/2}$ [Ber09]. SHA-2 and SHA-3 implementations are still considered safe in a post-quantum era [MVZJ18].

The NIST suggests that engineers expect to have a well functioning quantum computer within the next two decades [MCJ+16]. As a general rule of thumb predicting the urgency of the quantum threat, we use the formula $x + y > z$ from [Mos18], where x is the required lifetime of protecting security information and y the time required to implement quantum safe protocols. The last parameter z in the formula is the time left for a capable quantum computer to arrive. The article predicts that there is a 1/2 chance to break RSA-2048 in 2031 [Mos18]. In the case of travel documents, x is the lifetime of the keys of a passport, which is 13–15 years. The measurement of y is not even required to see that $13(+y) > 8$ at this point. This means that the integrity of currently issued travel documents is seriously at risk.

Our contributions:

- Benchmark of the NIST standardization candidates (signature algorithms) on Java level using the Bouncy Castle library in Java.
- Benchmark of the NIST standardization candidates used in X.509 certificates using the Bouncy Castle library in Java.
- A ICAO 9303 compliant PKI (excluding cryptography requirements) using post-quantum algorithms.
- A proof of concept quantum resistant eMRTD implementation of passive authentication.

2. Related work

Research of post-quantum cryptography is active and ongoing. The National Institute of Standards and Technology (NIST) is currently reviewing potential post-quantum algorithms to be standardized [AASA+19, AASA+20, AAC+22]. Also research has been done on the possibilities of potentially implementing these candidates in travel documents for specifically PA [PM20]. The focus of this research is to review the processes impacted by the quantum computer and to provide a proof of concept of the processes involved securing the eMRTD that will be impacted. The NIST started the process of investigating potential post-quantum resistant cryptographic algorithms in 2017, which has led to a total of 69 admitted algorithms to be reviewed for standardization [AASA+19]. At the end of the second round, 15 of investigated algorithms remain as standardization candidates including 7 finalists [AASA+20]. After the third round, there are four algorithms that will be standardized and another four algorithms will be evaluated further in the fourth round [AAC+22]. The candidates that will be standardized are CRYSTALS-KYBER, CRYSTALS-Dilithium, FALCON and SPHINCS$^+$ [AAC+22]. BIKE, Classic McEliece, HQC and SIKE are the other four algorithms that will be evaluated and reviewed further in the fourth round [AAC+22]. Candidates that are no longer considered after the third round are FrodoKEM, NTRU, NTRU Prime, Saber, G*e*MSS, Picnic and Rainbow [AAC+22]. The proposed post-quantum cryptographic algorithms are based on different principles in comparison with classical cryptosystems, and these algorithms include lattice-based, code-based, multivariate and hash-based cryptosystems [AAC+22]. Current developments indicate that the algorithms SIKE and Rainbow suffer from successful key recovery attacks, which reduces the cryptographic strength of these algorithms significantly [Beu22, CD22]. These attacks caused Rainbow to be dropped from further analysis by the NIST in the fourth round entirely [AAC+22]. During the research of potential safe algorithms for eMRTDs, it is important to keep track of similar developments on other proposed algorithms in order to establish a secure post-quantum solution.

There are also other signature schemes, such as XMSS [BDH11] and LMS [LM95]. These schemes are not being considered in the NIST PQC standardization program because these schemes are stateful and thus require careful implementation which make them impractical for general use [CAD+20]. Stateful hash-based algorithms make use of One-Time Signature Scheme (OTS) keys and an OTS may never be used to sign more than one message, which means that the used keys have to be monitored carefully [CAD+20]. If a key is reused accidentally, an attacker may use the two signatures to create a forgery [CAD+20]. This requires that the signing procedure should take place in a controlled secure environment to prevent OTS private keys being used more than once and this is generally considered a significant disadvantage over stateless cryptosystems [CAD+20]. Because the performance of XMSS is comparable to RSA [BDH11], it makes XMSS suitable to be used in resource constrained environments. This is shown for example in a study where the chain of trust uses a mix of CRYSTALS-Dilithium and XMSS in the chain of trust of a TLS X.509 PKI [MS22].

It is still hard to estimate the exact amount of qubits needed for performing a integer factorization or the measurement of a discrete logarithm using Shor's algorithm. A recent study that focused on improving existing techniques in order to factorize integers and computing discrete logarithms over finite fields roughly estimates the amount of (noisy) qubits required in order to break RSA-2048 in 8 hours is around 20 million [GE21]. The estimation is far from certain, but this estimation is already significantly lower than the 1 billion qubits estimated in earlier research [FMMC12]. A large proportion of the qubits are overhead which are needed to suppress the noise by performing error correction during calculations due to the fact that a quantum computer is very susceptible to noise compared to its classical counterpart [GE21, BKM+14]. So the power of a quantum computer does not merely depend on the amount of qubits available, it also depends on the ability to correct errors caused by noise and also the fact that quantum algorithms must be tailored in such a way that it makes more efficient use of the limited amount of qubits available. At the time of writing, IBM for example claims to have a quantum computer with a total of 433 qubits [CN22]. This is fortunately still far from the millions of qubits required in order to break modern cryptosystems. The quantum computer is not the ultimate solution for every mathematical problem available, but for particular problems such as integer factorization and discrete logarithms in finite fields however, it is a threat for the cryptosystems we use today.

Quantum computers have not gone unnoticed for passive authentication in travel documents. Recent research, based on the 7 NIST candidates for standardization, shows that X.509 certificates equipped with post-quantum algorithms can be used for passive authentication [PM20]. The CRYSTALS-Dilithium algorithm performed best in comparison with the other post-quantum algorithm for the purpose of passive authentication as long as the ICAO updates their minimum requirements for the chip of the eMRTD and the allowed algorithms to be used for passive authentication [PM20]. There have been some indirect developments by the progression of the NIST standardization process, as a result of which qTESLA and MQDSS have been dropped from the process after the second round [AASA+20] and the same occurred to Picnic and Rainbow in the third round [AAC+22]. Pradel and Mitchell [PM20] mention that their proof of concept was created by using an custom tailored OpenSSL PKI implementation for the CSCA and Document Signer certificates, and suggest to use Bouncy Castle[1] and JMRTD[2] instead for more compatibility with eMRTDs. Bouncy

[1] https://www.bouncycastle.org/

[2] https://jmrtd.org/

Castle is a free open source and light-weight cryptography API for Java and C#. At this date, Bouncy Castle has added all third round NIST PQC standardization candidates algorithms (see Table 10) to their low level API[3].

The eMRTD itself is also evolving over the years, adding more possibilities for border control. The second version of LDS (LDS2) adds new functionality such as digital Travel Records, Visa Records and Additional Biometrics applications and is backwards compatible with LDS 1.7 [ICA21c]. If used, these extra components also consume additional storage space on the chip. At the time of writing, there are chips available for eMRTDs containing up to 456 KB of EEPROM [4]. Due to the higher sizes of the keys and signatures in post-quantum algorithms [TLF+22, MK19], more capacity may be required for future chips in travel documents. Implementing post-quantum cryptography on devices with resource constraints poses a challenge and is an active topic of research. Small devices such as RFID-chips and embedded systems usually have low computing capacity, low memory and limited storage capacity. In a ubiquitously connected world, having devices using secure communication is important and these small devices also need to be secured against an upcoming quantum computer. Malina et al. provide an overview of the feasibility of post-quantum cryptography on small devices and conclude that lattice-based schemes such as New Hope and NTRU are promising for these small devices [MPD+18]. Another study performed by Marzougui and Krämer focused on Post-Quantum signature algorithms shows that qTESLA and XMSS perform well in constrained environments due to the small key and signature sizes and the fast validation speed [MK19]. More recently, a study shows that the lattice-based crypto scheme Dilithium is performing well in resource constrained environments based on energy consumption unlike SPHINCS+ [TDF+23].

A related protocol which involves digital certificates and is vulnerable for quantum computers is TLS, particularly in constrained environments. Tasopoulos et al. outline the changes required in the architecture of TLS 1.3 in order to integrate the NIST 4th round post-quantum cryptosystems on constrained embedded devices [TLF+22]. Similarly, Gonzales and Wiggers compare KEMTLS, an alternative TLS handshake protocol, with TLS 1.3 and conclude that KEMTLS uses less memory than the TLS 1.3 implementation [GW22]. In the field there are more examples of integration of PQ crypto primitives into TLS with the focus on constrained environments, such as [BSKNS20].

A recent study of European Union Agency for Cybersecurity (ENISA) shows an overview

[3]See Bouncy Castle release notes for version 2.4.3 (https://www.bouncycastle.org/releasenotes.html)

[4]In this case programmable Java Cards which can also be used for simulating eMRTD chip behaviour: https://www.infineon.com/cms/en/product/security-smart-card-solutions/secora-security-solutions/secora-id-security-solutions/

of the various post-quantum cryptosystems with their advantages and disadvantages [BHL22]. Their study shows that hash based signatures have small public keys and larger signatures whereas multivariate-based cryptosystems have large public keys and small signatures [BHL22]. The size of the public keys and signatures of lattice-based cryptosystems are inbetween the multivariate-based and hash-based cryptosystems according to this study [BHL22].

3. Post-quantum threats to the eMRTD security mechanisms

Quantum computing is a risk for the processes involved in security an eMRTD. In order to pinpoint the vulnerabilities of these protocols, it is important to analyse the different security protocols of the eMRTD individually. In order to perform such analysis, we focus on the breakability of the cryptosystems by quantum computers used within the different protocols, the result gained by such a breach and the scope of the result gained by the breach. This will give a direction to the intended research on reducing the threat of quantum computers on the security of the eMRTD. In the following sections we will explain the protocol as defined in ICAO 9303 (part 11) [ICA21c] and we will outline the risks for the discussed protocol.

3.1. Basic Access Control (BAC)

BAC uses the two key variant of Triple DES (3DES) as block cipher with its keys derived from the Machine Readable Zone (MRZ) in order to access the less sensitive data groups of the eMRTD and to start secure messaging between the reader and the chip [ICA21c]. BAC is initiated by the reader (IFD) and requests a challenge (nonce) from the chip (IC) as shown in Figure 1 [ICA21c]. After receiving the RND.IC (nonce) from the chip, the terminal also generates 2 random values, an 8 bytes RND.IFD and a 16 bytes K.IFD and concatenates this with the nonce received from the chip into S [ICA21c]. This value will be encrypted with 3DES by using K_{Enc} and results in E_{IFD}. A MAC will be created in a similar manner, but in this case with K_{MAC} and yields M_{IFD}. The K_{Enc} and K_{MAC} are derived from the MRZ [ICA21c], as shown in Figure 2. The figure shows K_a and K_b, because BAC uses a two key 3DES [ICA21c]. The E_{IFD} and M_{IFD} are sent to the chip by issuing a *External Authentication* command [ICA21c]. The chip then checks the received checksum and generates K.IC when all required checks are performed successfully [ICA21c]. The chip will concatenate the received nonces (RND.IC and RND.IFD) and the K.IC into R [ICA21c]. The result of this process is encrypted with K_{Enc} into E_{IC} and accompanied with a MAC M_{IC} enciphered with K_{MAC} [ICA21c]. The values E_{IC} and M_{IC} are sent back to the reader which verifies the received data accordingly [ICA21c]. Both sides can now derive the session keys KS_{Enc} and KS_{MAC} and the Send Sequence Counter (SSC) [ICA21c]. The SSC for BAC is based on concatenation of the 4 least significant bytes of the RND.IC and RND.IFD [ICA21c].

The use of 3DES (or officially TDEA) has been discouraged by the NIST since 2017 [oST17] and 3DES will become deprecated after 2023 [BR19], even though 3DES has not been officially broken yet. Aside from the 3DES deprecation, BAC also suffers from a low cryptographic strength due to its limited randomness in the generated

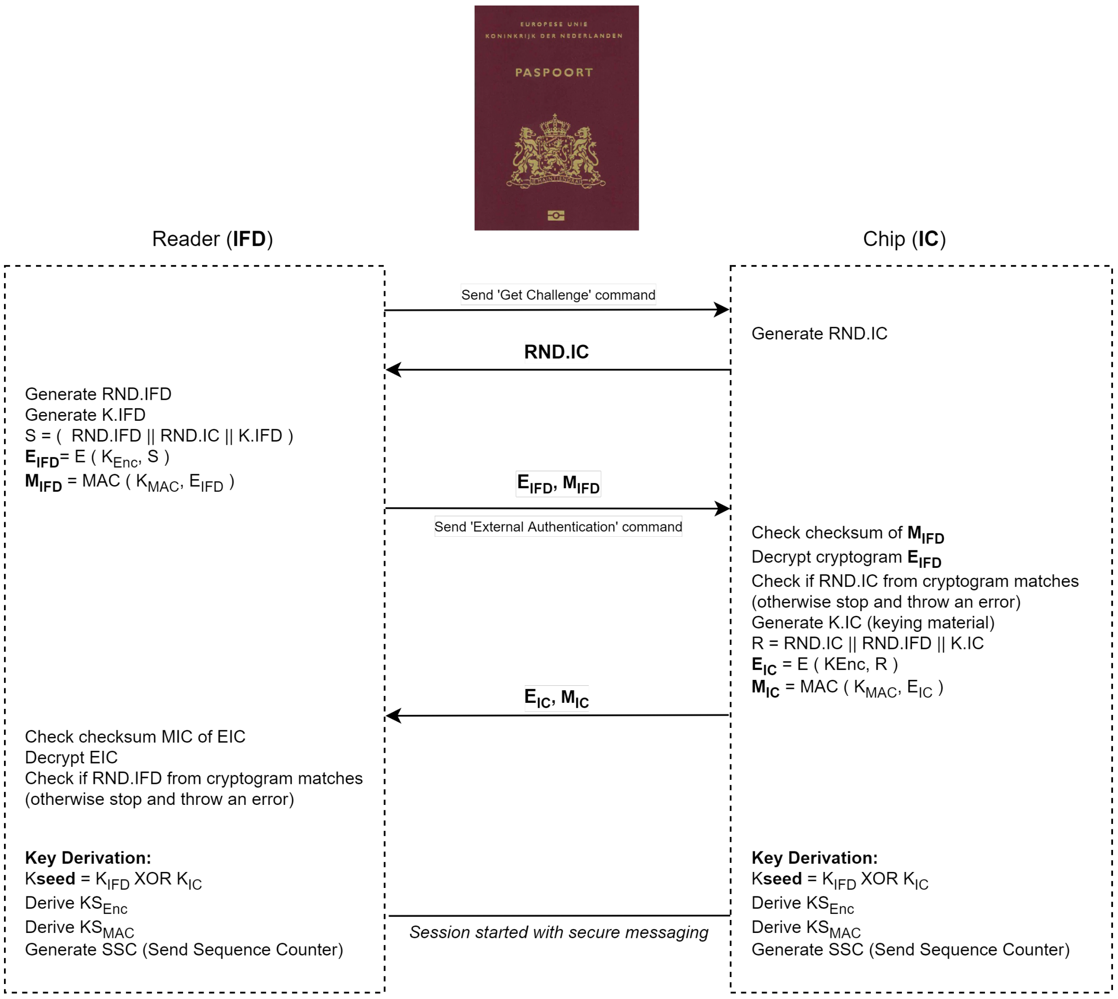

EUROPESE UNIE
KONINKRIJK DER NEDERLANDEN
PASPOORT
Reader (**IFD**)
Chip (**IC**)
Send 'Get Challenge' command
Generate RND.IC
RND.IC
Generate RND.IFD
Generate K.IFD
S = (RND.IFD || RND.IC || K.IFD)
$\mathbf{E_{IFD}}$= E (K_{Enc}, S)
$\mathbf{M_{IFD}}$ = MAC (K_{MAC}, E_{IFD})
$\mathbf{E_{IFD}, M_{IFD}}$
Send 'External Authentication' command
Check checksum of $\mathbf{M_{IFD}}$
Decrypt cryptogram $\mathbf{E_{IFD}}$
Check if RND.IC from cryptogram matches (otherwise stop and throw an error)
Generate K.IC (keying material)
R = RND.IC || RND.IFD || K.IC
$\mathbf{E_{IC}}$ = E (KEnc, R)
$\mathbf{M_{IC}}$ = MAC (K_{MAC}, E_{IC})
$\mathbf{E_{IC}, M_{IC}}$
Check checksum MIC of EIC
Decrypt EIC
Check if RND.IFD from cryptogram matches (otherwise stop and throw an error)
Key Derivation:
Kseed = K_{IFD} XOR K_{IC}
Derive KS_{Enc}
Derive KS_{MAC}
Generate SSC (Send Sequence Counter)
Session started with secure messaging
Key Derivation:
Kseed = K_{IFD} XOR K_{IC}
Derive KS_{Enc}
Derive KS_{MAC}
Generate SSC (Send Sequence Counter)

Figure 1: Steps for performing BAC to access the data groups and to establish secure messaging. Derived from the steps required in ICAO 9303 part 11 [ICA21c].

keys. A new protocol called PACE was designed to overcome this limitation [ICA21c]. The usage of BAC has been suspended by the ICAO for new passports since 2018, but the terminals should still support BAC when the document has no PACE implementation available [ICA21c]. It is still possible to encounter documents supporting only BAC, because the PACE-only requirement has been issued since 2018 and eMRTDs can have a validity period of ten to fifteen years [ICA21c]. During the key derivation process, as shown in Figure 2, BAC makes use of the SHA-1 hashing algorithm, which is already deprecated to be used by the NIST back in 2011 [BR19]. Due to the fact that BAC is being phased out for eMRTDs and the fact that 3DES could only be affected by the square root speed up achieved by Grover's algorithm [Gro96], the risk to the security of the eMRTD is low.

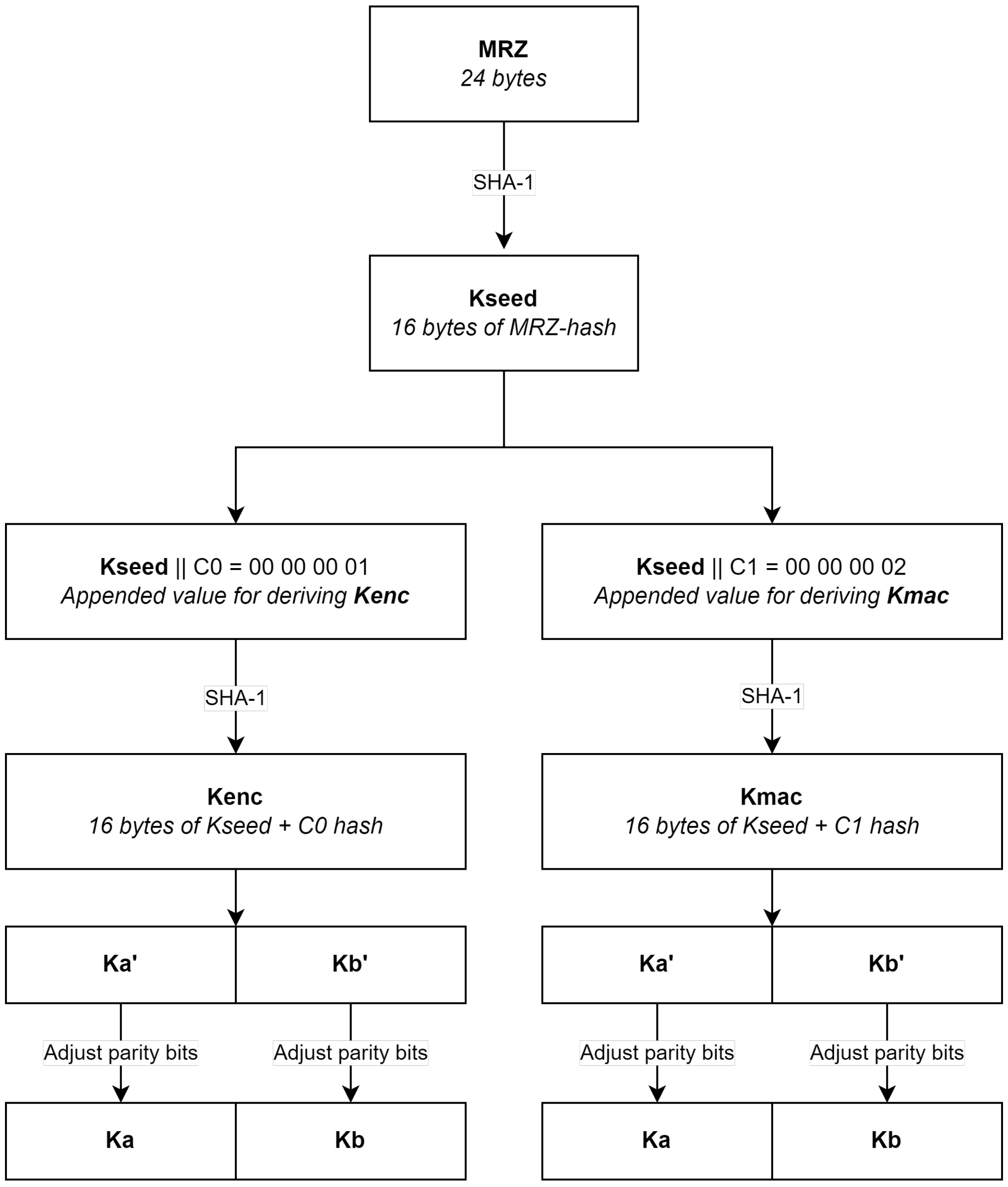

MRZ
24 bytes
SHA-1
Kseed
16 bytes of MRZ-hash
Kseed || C0 = 00 00 00 01
Appended value for deriving Kenc
Kseed || C1 = 00 00 00 02
Appended value for deriving Kmac
SHA-1
SHA-1
Kenc
16 bytes of Kseed + C0 hash
Kmac
16 bytes of Kseed + C1 hash
Ka'
Kb'
Ka'
Kb'
Adjust parity bits
Adjust parity bits
Adjust parity bits
Adjust parity bits
Ka
Kb
Ka
Kb

Figure 2: The key derivation process within BAC as defined in ICAO 9303 part 11 [ICA21c].

3.2. Password Authenticated Connection Establishment (PACE)

PACE is a key agreement protocol using Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) that enables password authentication and establishes secure communication messaging between the chip and the reader based on weak passwords [ICA21c]. It is more secure than BAC, because the created session keys are independent of the entropy of the password strength [ICA21c].

PACE starts by accessing the EF.Cardaccess on the chip, if this elementary file does not exist, PACE cannot start, because this file describes which algorithm and format is used for establishing secure messaging [ICA21c]. After reading the security parameters, the reader issues a 'Set Authentication Template' command which instructs the chip to start the PACE protocol [ICA21c]. The chip acknowledges this instruction by sending a reply to the reader, and after this step the reader starts sending several 'General Authenticate' commands to follow the required steps of PACE as shown in Figure 3 [ICA21c]. In the first step the reader requests a nonce from the chip, which is encrypted with the key derivation function KDF_{π} by using the shared password π, which is the MRZ (required) or CAN (optional) [ICA21c]. The key derivation function derives keys for PACE by using π which results in the key K_{π} [ICA21c]. The reader decrypts z containing the nonce by using the shared password π ($K_{\pi} = KDF_{\pi}(\pi)$) [ICA21c]. Depending on the used mapping of the nonce, the reader and the chip exchange additional data (displayed as conditional steps in Figure 3) [ICA21c]. Mapping is a cryptographic mechanism to map a nonce to a pseudo random number generator to be used in the asymmetric cryptosystems [ICA21c]. These mappings are *Generic Mapping* (based on a DH-key agreement), *Integrated Mapping* (direct mapping of field element to the cryptographic group) and *Chip Authentication Mapping* (used for Chip Authentication) [ICA21c]. Subsequently, the reader and the chip compute the ephemeral domain parameters [ICA21c]. After computing the domain parameters, the reader and the chip perform a Diffie-Hellman key exchange using the domain parameters D to generate the shared key K as shown in Figure 3 [ICA21c]. After generating the shared key K, the chip and the reader derive the session keys KS_{MAC} and KS_{Enc} by using the corresponding key derivation functions (KDF) [ICA21c]. This process is similar to the key derivation process during BAC. As a final step, the reader and the chip exchange and verify the authentication token, which is to verify that both sides use the proper K_{π} [ICA21c]. If the chip supports Chip Authentication then there will be an extra step, which will be discussed in more detail in the CA section of this chapter. From this point, access has been granted to the data groups and secure messaging has been established.

Similar to BAC, PACE provides security by restricting access to the less privacy-sensitive data groups and prevents eavesdropping on the communication between the chip and the reader. As mentioned earlier, there are three mappings for establishing a shared key for secure communication between the reader and the chip. A limited set of ciphers and configurations are allowed depending on the chosen mapping for PACE. The possible configurations for each mapping using DH are shown in Table 1. As mentioned earlier, PACE also supports the use of ECDH and has a similar predefined set of ciphers and configurations which are allowed to be used. The allowed configurations for ECDH can be seen in Table 2, along with the ciphers and the key length. These ciphers share commonalities with those of the PACE-DH. For chips

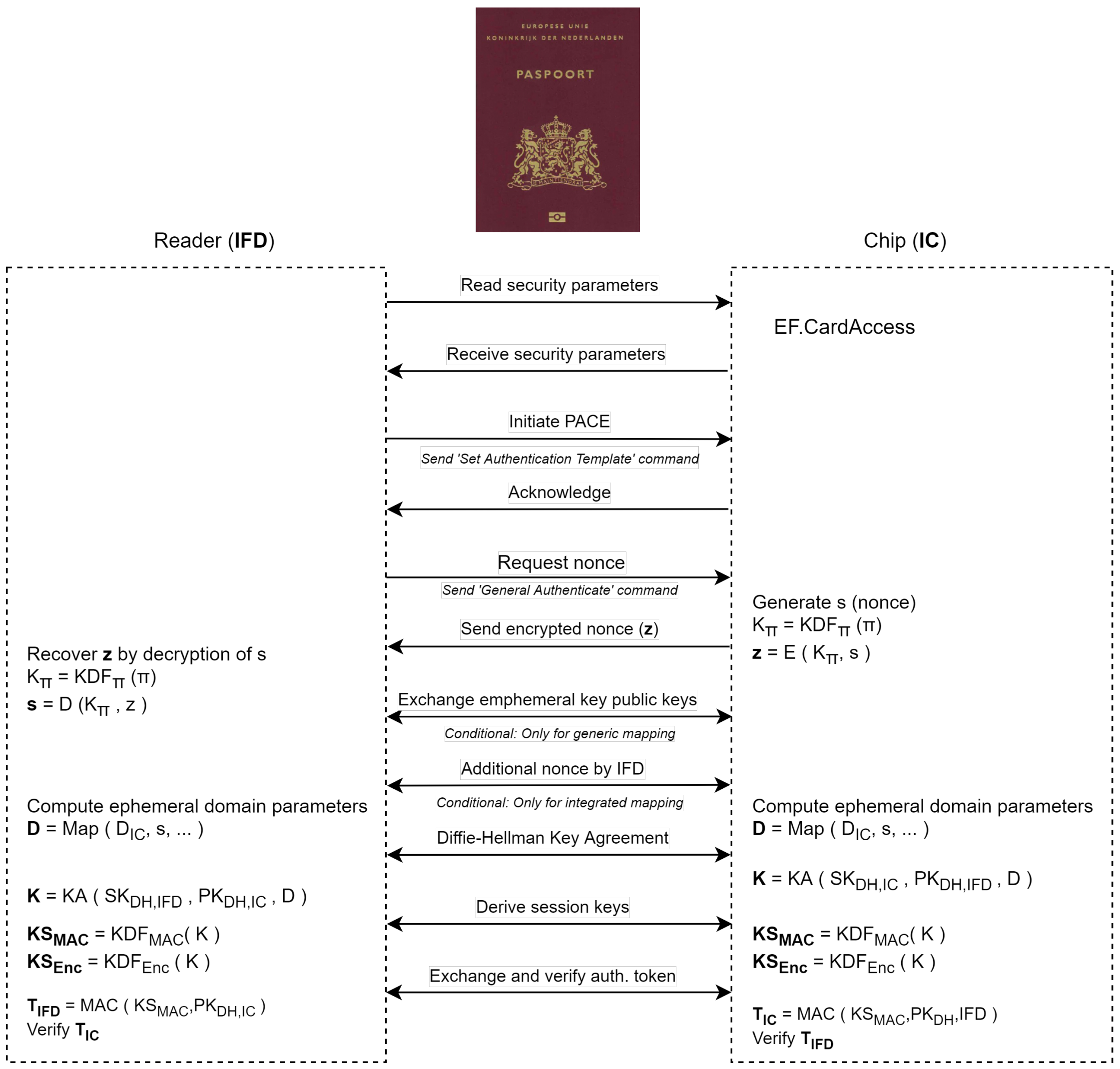

EUROPESE UNIE
KONINKRIJK DER NEDERLANDEN
PASPOORT
Reader (**IFD**)
Chip (**IC**)
Read security parameters
EF.CardAccess
Receive security parameters
Initiate PACE
Send 'Set Authentication Template' command
Acknowledge
Request nonce
Send 'General Authenticate' command
Generate s (nonce)
K_{π} = KDF_{π} (π)
z = E (K_{π}, s)
Send encrypted nonce (**z**)
Recover **z** by decryption of s
K_{π} = KDF_{π} (π)
s = D (K_{π} , z)
Exchange emphemeral key public keys
Conditional: Only for generic mapping
Additional nonce by IFD
Conditional: Only for integrated mapping
Compute ephemeral domain parameters
D = Map (D_{IC}, s, ...)
Compute ephemeral domain parameters
D = Map (D_{IC}, s, ...)
Diffie-Hellman Key Agreement
K = KA ($SK_{DH,IFD}$, $PK_{DH,IC}$, D)
K = KA ($SK_{DH,IC}$, $PK_{DH,IFD}$, D)
Derive session keys
$\mathbf{KS_{MAC}}$ = KDF_{MAC}(K)
$\mathbf{KS_{Enc}}$ = KDF_{Enc} (K)
$\mathbf{KS_{MAC}}$ = KDF_{MAC}(K)
$\mathbf{KS_{Enc}}$ = KDF_{Enc} (K)
Exchange and verify auth. token
$\mathbf{T_{IFD}}$ = MAC (KS_{MAC},$PK_{DH,IC}$)
Verify $\mathbf{T_{IC}}$
$\mathbf{T_{IC}}$ = MAC (KS_{MAC},PK_{DH},IFD)
Verify $\mathbf{T_{IFD}}$

Figure 3: Process flow of the PACE protocol (summarized) derived from [ICA21c]

supporting CA, the allowed ciphers and configurations are limited to ECDH using AES of at least 128 bits [ICA21c]. Important to notice in both tables is that in a post-quantum era, the shown key strengths would be halved.

According to both sets of specifications, PACE makes use of 3DES and AES ciphers which are symmetrical encryption algorithms and could be affected by Grover's algorithm in a post-quantum era. However, AES256 is still considered secure in a post-quantum era [RMYK17], because the root speed up achieved by Grover's algorithm [Gro96] would drop the strength of AES256 to just 128 bits, which is still considered strong. AES is not the only cipher allowed for PACE, the latest specifications of the ICAO also imply that 3DES is also still allowed in some circumstances [ICA21c]. As

OID	Sym. Cipher	Key length
id-PACE-DH-GM-3DES-CBC-CBC	3DES	112
id-PACE-DH-GM-AES-CBC-CMAC-128	AES	128
id-PACE-DH-GM-AES-CBC-CMAC-192	AES	192
id-PACE-DH-GM-AES-CBC-CMAC-256	AES	256
id-PACE-DH-IM-3DES-CBC-CBC	3DES	112
id-PACE-DH-IM-AES-CBC-CMAC-128	AES	128
id-PACE-DH-IM-AES-CBC-CMAC-192	AES	192
id-PACE-DH-IM-AES-CBC-CMAC-256	AES	256

Table 1: Algorithms and formats allowed for PACE-DH (simplified) [ICA21c]

OID	Sym. Cipher	Key length
id-PACE-ECDH-GM-3DES-CBC-CBC	3DES	112
id-PACE-ECDH-GM-AES-CBC-CMAC-128	AES	128
id-PACE-ECDH-GM-AES-CBC-CMAC-192	AES	192
id-PACE-ECDH-GM-AES-CBC-CMAC-256	AES	256
id-PACE-ECDH-IM-3DES-CBC-CBC	3DES	112
id-PACE-ECDH-IM-AES-CBC-CMAC-128	AES	128
id-PACE-ECDH-IM-AES-CBC-CMAC-192	AES	192
id-PACE-ECDH-IM-AES-CBC-CMAC-256	AES	256
id-PACE-ECDH-CAM-AES-CBC-CMAC-128	AES	128
id-PACE-ECDH-CAM-AES-CBC-CMAC-192	AES	192
id-PACE-ECDH-CAM-AES-CBC-CMAC-256	AES	256

Table 2: Algorithms and formats allowed for PACE-ECDH (simplified) [ICA21c]

mentioned earlier, the NIST has decided to deprecate the 3DES algorithm back in 2017 [oST17], so the use of 3DES is discouraged. Even though 3DES is being deprecated, there are no signs yet that 3DES will be broken in the near future using quantum computers. In an overview of quantum threats on modern cryptography, the threat on symmetrical cryptosystems is even considered as minor [MVZJ18].

The agreement of a shared key however is more interesting, because DH and ECDH are based on asymmetric cryptography. PACE uses DH and ECDH for establishing secure session keys between the reader and the chip [ICA21c]. These algorithms are vulnerable to Shor's algorithm [Sho97] due to the solvability of the discrete logarithm problem where DH depends on. In a post-quantum era it could be possible to derive the session keys by simply observing the communication and obtaining the public elements of the DH key exchange. In the context of the eMRTD this risk decreases because the key exchange and the session keys are no longer valid after the passport holder details have been read by the reader. Also the passport holder details can be read by anyone in possession of the physical travel document, so the effort required to electronically access the document would be significantly higher than the effort to just stealing the physical travel document instead. Due to the fact that the established keys are used for one session only and that the overall gains are only related to potential privacy issues, the impact on the eMRTD security of PACE being broken by a quantum computer is low.

3.3. Passive Authentication (PA)

As mentioned earlier, passive authentication is the core of a travel document as its structure is designed to prevent unauthorized changes of the data within the chip. During the production of the chip, the data on the holder's page, such as personal information, is placed in the data groups of the Logical Data Structure (LDS) within the eMRTD. The LDS acts as a structured storage and can be compared to a file system with a different protocol describing how to read and write to this file system. The structure of the data groups within the LDS is displayed in Figure 4.

In order to protect the data within the data groups, the hashes of these data groups are calculated and electronically signed by a Document Signer (DS) as a part of the Public Key Infrastructure (PKI) for passive authentication [ICA21d]. The hashes and the signature of the hashes are placed in the Document Security Object (EF.SOD) of the LDS [ICA21b]. The document signer has been issued by the certificate authority called the Country Signing Certificate Authority (CSCA) and is the root of the PKI structure for passive authentication [ICA21d]. These are X.509 certificates with a strict set of mandatory properties in order to be used for passive authentication and are defined in ICAO 9303 part 12 [ICA21d]. During verification of a travel document, the signatures of the data group hashes will be validated using the public key of the document signer [ICA21d]. If any unauthorized change has been made in any of the data groups, the involved hashes will be different and the signature of these hashes will not match. This process is visualized in Figure 5.

Every country has exactly one CSCA [ICA21d]. In order to establish a chain of trust between the CSCA and the Document Signer (DS) during a border inspection, each CSCA certificate that is trusted in a country needs to be distributed towards the readers at the border. Trusted CSCA certificates are located in a nation's National Public Key Directory (NPKD) and is used in the infrastructure to distribute the CSCA certificates towards the terminals at the country's borders [ICA21d]. For the CSCA, each country may decide to use RSA, DSA or ECDSA as algorithm for the CSCA and signing certificate keys as specified by the eighth version of the ICAO 9303 [ICA21d]. The validity of a CSCA differs per country, but the public key of the CSCA must be valid for at least 13–15 years and the usage of the CSCA private key is limited to 3-5 years

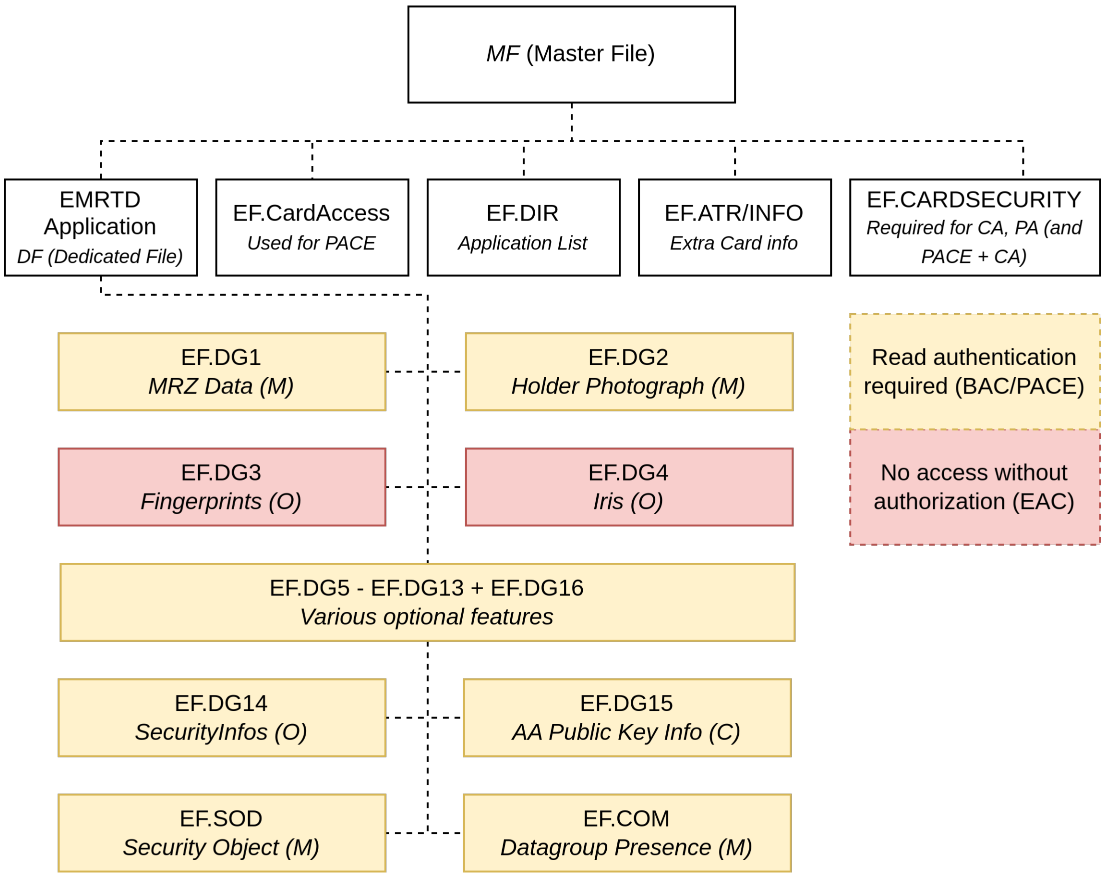


Figure 4: Structure of LDS within the eMRTD derived from [ICA21b]

[ICA21d]. RSA, DSA and ECDSA are asymmetric algorithms and are vulnerable to Shor's algorithm using quantum computers [Sho97]. RSA, DSA and ECDSA also make use of hashing algorithms. As discussed earlier, they are vulnerable to a collision search speedup [MVZJ18]. The allowed hash algorithms for passive authentication (CSCA and DS certificates) are SHA-224, SHA-256, SHA-384 and SHA-512 [ICA21d]. These hash algorithms are still considered safe in a quantum era [MVZJ18].

The CSCA is responsible for signing document signers in order to guarantee the integrity of a travel document. If a quantum computer would be established in the next decade with enough computational power to dissolve the mathematical problems that RSA, DSA and ECDSA depend on, then the security of the travel document could be at risk, because it is possible to obtain the private key of a CSCA or DS. This would allow governments with malicious intent or criminal organizations to enroll fraudulent travel documents which would be validated as genuine at an automated border inspection. Due to the widespread consequences of a potential breach of these asymmetric algorithms, the impact on the security of the eMRTD is considered high.

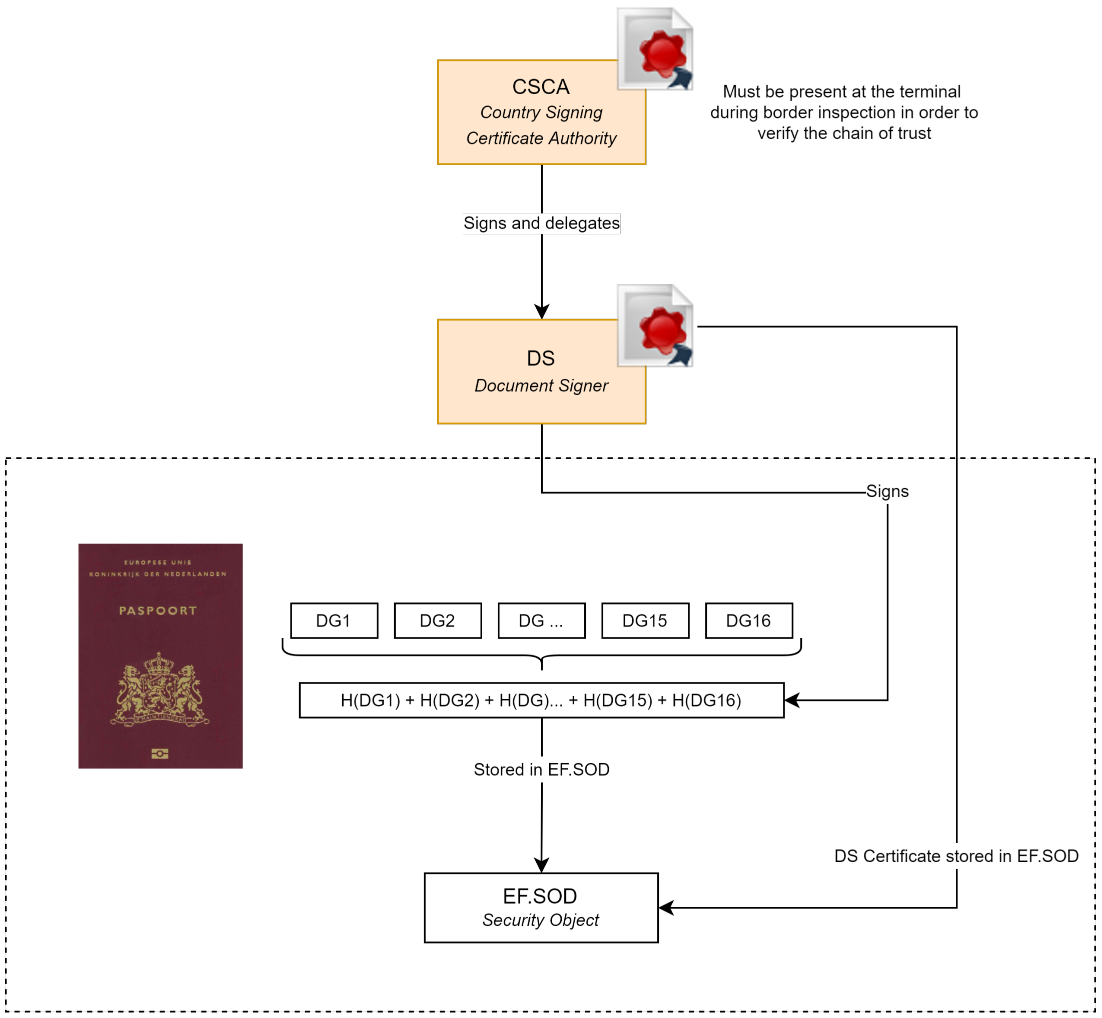

CSCA
Country Signing Certificate Authority
Must be present at the terminal during border inspection in order to verify the chain of trust
Signs and delegates
DS
Document Signer
Signs
EUROPESE UNIE
KONINKRIJK DER NEDERLANDEN
PASPOORT
DG1
DG2
DG ...
DG15
DG16
H(DG1) + H(DG2) + H(DG)... + H(DG15) + H(DG16)
Stored in EF.SOD
DS Certificate stored in EF.SOD
EF.SOD
Security Object

Figure 5: Passive authentication to prevent unauthorized changes within the passport chip.

3.4. Active Authentication (AA)

Active Authentication (AA) is an optional protocol that authenticates the eMRTD chip by using a challenge-response mechanism in order to prevent document cloning [ICA21c]. Cloning is a full copy of the contents of the chip to another chip to be used in a counterfeit document for example. The only thing that cannot be copied is the private key located in the secure memory of the chip, which can only be accessed internally by the chip itself [ICA21c]. The public key corresponding with the private key is is stored in data group 15 in order to perform AA [ICA21c]. Data group 15 contains a SubjectPublicKeyInfo object, as defined in RFC-5280 [BSP^{+}08], and describes the public key alongside the details of the algorithms used. During AA, the reader sends a nonce to the chip which is signed by the private key of the chip [ICA21c]. This signed nonce is sent back and the reader verifies the signature by using the public key present in data group 15 [ICA21c]. Depending of the algorithm used, the process flow of active authentication is described in Figure 6. The communication flow is the same for RSA and ECDSA, but when using RSA, a few additional steps are required for active authentication, such as adding a trailer depending on the hashing algorithm used and an additional nonce generated by the chip [ICA21c].

There are some privacy concerns for using AA however, because AA may allow tracking of the passport due to the challenge-response nature of the protocol especially when there is no BAC security on a passport [BSI15]. Instead of using AA, it is also possible to use CA, because CA also proves ownership of the original private key of the chip. RSA and ECDSA are the only two algorithms allowed to compose the key pairs for active authentication in travel documents [ICA21c]. These algorithms are allowed to be used in combination with the hash functions as indicated by ICAO 9303 specifications shown in table 3. An important note is that SHA-1 is only allowed to be used in combination with RSA [ICA21c].

Similar to PA, AA makes also use of asymmetric cryptography algorithms such as RSA and ECDSA which are vulnerable to Shor's algorithm in a post-quantum era [Sho97]. Hash algorithms are also a component of active authentication and only SHA-1[5], SHA-224, SHA-256, SHA-384 and SHA-512 are allowed to be used as hash algorithm for active authentication [ICA21c]. Aside from SHA-1, since this algorithm is already considered insecure, the other mentioned hash algorithms are still considered safe in a post-quantum era [MVZJ18].

If the private key stored in the secure memory can be derived using quantum computers, and the key can be additionally copied to the cloned document. Since the cloned document now has the access to the private key, it is now able to perform active authentication without triggering alarms to the reader. Cloning however has limitations, because the data itself has not been altered due to the protection of passive authentication. This means that the holder data has been unchanged, such as the passport photo and the age of the passport bearer. The cloned passport is then only useful for persons who visually match the cloned passport. Also, the use of active authentication in an eMRTD is optional [ICA21c]. If quantum computers are powerful enough to break RSA and ECDSA it would have an impact on active authentication. However the impact on the security of the eMRTD is limited, because the data itself cannot be altered and AA is an optional security feature. Therefore the impact on the overall eMRTD security will be low.

[5]Only to be used with RSA for interoperability reasons, since it is already considered insecure [ICA21c].

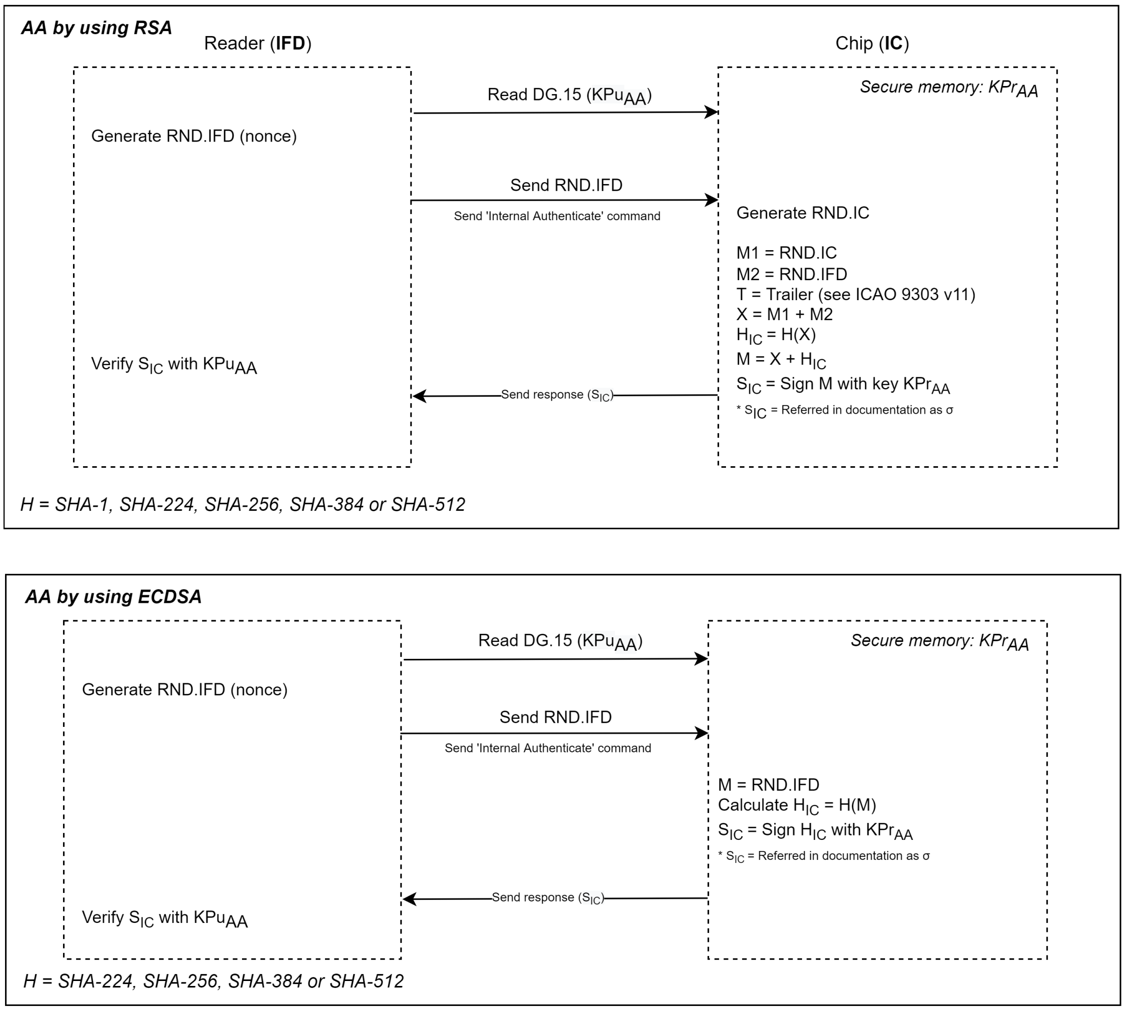

AA by using RSA
Reader (IFD)
Chip (IC)
Read DG.15 (KPu_{AA})
Secure memory: KPr_{AA}
Generate RND.IFD (nonce)
Send RND.IFD
Send 'Internal Authenticate' command
Generate RND.IC
M1 = RND.IC
M2 = RND.IFD
T = Trailer (see ICAO 9303 v11)
X = M1 + M2
H_{IC} = H(X)
M = X + H_{IC}
S_{IC} = Sign M with key KPr_{AA}
* S_{IC} = Referred in documentation as σ
Verify S_{IC} with KPu_{AA}
Send response (S_{IC})
H = SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512
AA by using ECDSA
Read DG.15 (KPu_{AA})
Secure memory: KPr_{AA}
Generate RND.IFD (nonce)
Send RND.IFD
Send 'Internal Authenticate' command
M = RND.IFD
Calculate H_{IC} = H(M)
S_{IC} = Sign H_{IC} with KPr_{AA}
* S_{IC} = Referred in documentation as σ
Send response (S_{IC})
Verify S_{IC} with KPu_{AA}
H = SHA-224, SHA-256, SHA-384 or SHA-512

Figure 6: Active Authentication process in an eMRTD by using RSA or ECDSA, derived from [ICA21c].

3.5. Chip Authentication (CA)

The main purpose of chip authentication is to prove that the chip is not cloned and establishes strong session keys for the communication between the terminal and the chip [ICA21c]. In contrast to AA, CA performs a DH or ECDH key exchange instead of a challenge-response based mechanism to prove that the chip is not cloned and at

Algorithm	Supported hash functions
RSA	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
ECDSA	SHA-224, SHA-256, SHA-384, SHA-512

Table 3: Supported hash functions for RSA and ECDSA [ICA21c]

the same time establishes stronger keys for the communication between the reader and the chip [ICA21c]. The protocol dictates that the inspection system reads the public key of the chip located in data group 14 and the terminal generates a public and private key pair [ICA21c]. The public key generated by the inspection system is sent to the chip and the terminal and chip derive a shared key by performing a Diffie Hellman key agreement [ICA21c]. The difference with a regular DH key exchange is that the private and the public keys of the chip are static and the private key is located in the secure memory of the chip [ICA21c]. Figure 7 provides a general overview of the chip authentication protocol. The chip authentication protocol does not have the privacy issue present in AA according to the German Bundesamt für Sicherheit in der Informationstechnik (BSI), because the private key of the chip is not signing arbitrary data from the terminal [BSI15]. An important note for CA is that passive authentication must be performed in order to verify that the public key present in data group 14 has not been altered in any way [ICA21c]. Chip authentication is now mandatory for European documents supporting EAC [ICA21c].

CA involves asymmetric cryptography for establishing a shared key and symmetrical cryptography for the secure channel according to Table 4 and Table 5 derived from the ICAO 9303 specifications. In the previous sections, we have seen that symmetrical ciphers such as 3DES and AES, which are also used in CA, are only vulnerable to Grover's square root speedup in a post-quantum era [Gro96]. The asymmetric parts of the protocol, such as the key exchange, are vulnerable for Shor's algorithm [Sho97]. But also for chip authentication, the ephemeral keys used during the session are not used anymore after the session has been closed. Additionally, CA is only an obligation for European documents. Taking these factors into account, the security impact on the eMRTD as a whole is low.

OID	Cipher	Key Length	Secure Messaging
id-CA-DH-3DES-CBC-CBC	3DES	112	CBC / CBC
id-CA-DH-AES-CBC-CMAC-128	AES	128	CBC / CMAC
id-CA-DH-AES-CBC-CMAC-192	AES	192	CBC / CMAC
id-CA-DH-AES-CBC-CMAC-256	AES	256	CBC / CMAC

Table 4: Supported algorithms and key lengths for CA with DH [ICA21c]

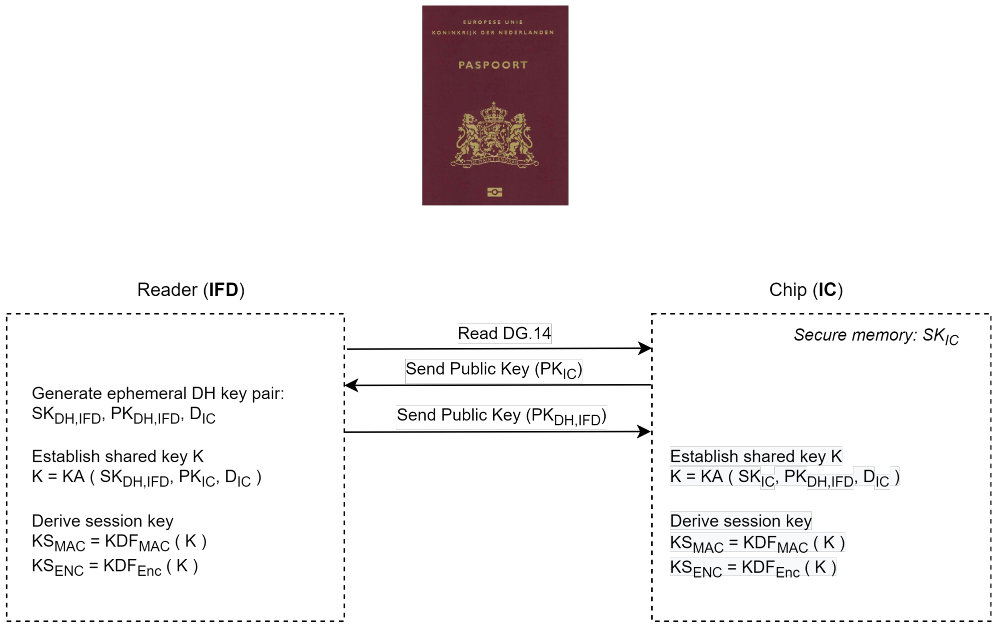

EUROPESE UNIE
KONINKRIJK DER NEDERLANDEN
PASPOORT
Reader (IFD)
Chip (IC)
Secure memory: SK_{IC}
Read DG.14
Send Public Key (PK_{IC})
Send Public Key ($PK_{DH,IFD}$)
Generate ephemeral DH key pair:
$SK_{DH,IFD}$, $PK_{DH,IFD}$, D_{IC}
Establish shared key K
K = KA ($SK_{DH,IFD}$, PK_{IC}, D_{IC})
Derive session key
KS_{MAC} = KDF_{MAC} (K)
KS_{ENC} = KDF_{Enc} (K)
Establish shared key K
K = KA (SK_{IC}, $PK_{DH,IFD}$, D_{IC})
Derive session key
KS_{MAC} = KDF_{MAC} (K)
KS_{ENC} = KDF_{Enc} (K)

Figure 7: The Chip Authentication process within the eMRTD for establishing stronger keys by using a DH key agreement (shortened as KA in the overview) using predetermined domain parameters (D). Process flow and protocol information derived from [ICA21c].

OID	Cipher	Key Length	Secure Messaging
id-CA-ECDH-3DES-CBC-CBC	3DES	112	CBC / CBC
id-CA-ECDH-AES-CBC-CMAC-128	AES	128	CBC / CMAC
id-CA-ECDH-AES-CBC-CMAC-192	AES	192	CBC / CMAC
id-CA-ECDH-AES-CBC-CMAC-256	AES	256	CBC / CMAC

Table 5: Supported algorithms and key lengths for CA with ECDH [ICA21c]

3.6. Terminal Authentication (TA)

If the chip supports Extended Access Control (EAC), which is mandatory for all EU passports, then Terminal Authentication (TA) has to be executed after performing PACE with CA mapping [ICA21c]. In TA, the chip verifies if the reader is allowed to access the privacy-sensitive data groups [ICA21c]. A separate PKI based on Card Verifiable Certificates (CVC) needs to be established for TA in order to perform this authentication process [ICA21c]. This PKI consists of a Country Verifying Certificate Authority (CVCA), Document Verifying Certificate Authority (DVCA) and Inspection System (IS) certificate chain and the CVCA certificate is stored in the chip during production [ICA21c]. When performing TA at a border inspection, the reader sends its inspection system certificate with corresponding access permissions and the rest of the CV certificate chain above to the chip [ICA21c]. The chip uses the certificate chain to verify if all of the provided certificates are correct and if the terminal possesses the correct access permissions [ICA21c]. To ensure that the terminal is authentic, the chip also sends a challenge to the reader which needs to be signed by the private key of the terminal [ICA21c]. The chip itself validates challenge by using the public key of the inspection system, and upon success, the chip provides access to the privacy-sensitive parts of the eMRTD [ICA21c]. This process is shown visually in Figure 8.

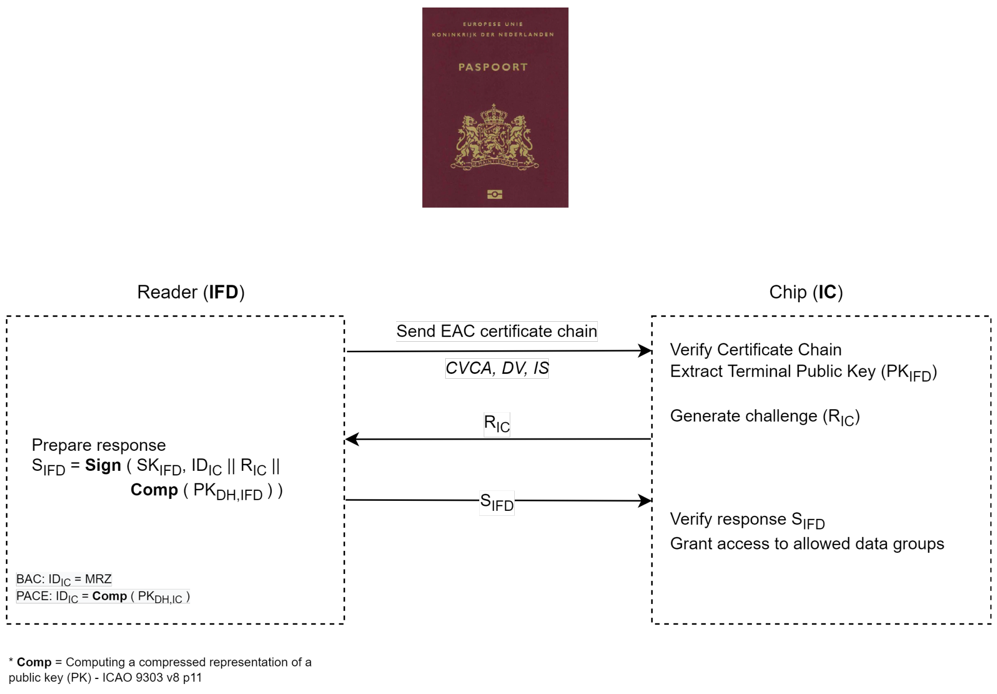

EUROPESE UNIE
KONINKRIJK DER NEDERLANDEN
PASPOORT
Reader (**IFD**)
Chip (**IC**)
Send EAC certificate chain
CVCA, DV, IS
Verify Certificate Chain
Extract Terminal Public Key (PK_{IFD})
Generate challenge (R_{IC})
R_{IC}
Prepare response
S_{IFD} = **Sign** (SK_{IFD}, ID_{IC} || R_{IC} || **Comp** ($PK_{DH,IFD}$))
S_{IFD}
Verify response S_{IFD}
Grant access to allowed data groups
BAC: ID_{IC} = MRZ
PACE: ID_{IC} = **Comp** ($PK_{DH,IC}$)
* **Comp** = Computing a compressed representation of a public key (PK) - ICAO 9303 v8 p11

Figure 8: Terminal authentication (simplified) with the protocol specifications derived from [ICA21c].

Similar to the other eMRTD protocols, there are a limited set of ciphers and configurations which are allowed to be used. For TA, the allowed ciphers are either RSA or ECDSA in predefined configurations as shown in Table 6 and Table 7. As seen earlier, RSA and ECDSA are asymmetric ciphers which are vulnerable to Shor's algorithm [Sho97]. When a quantum computer with enough computation power is available, it would be possible to issue terminal certificates with the access permissions to data group 3 and 4. This would compromise the privacy-sensitive data if present in the corresponding data groups. RSA and ECDSA also use hash algorithms in their implementations. SHA-224, SHA-256, SHA-384 and SHA-512 are allowed hash algorithms for TA [ICA21c]. As mentioned earlier, these algorithms are considered safe in a post-quantum era [MVZJ18]. Additionally, TA is not a mandatory requirement outside the European Union, so countries may decide individually to use TA for securing the privacy-sensitive data [ICA21c]. Also even if access is obtained to the data groups, the risk is limited to a privacy issue for the holder of the travel document. In order to fully forge a passport including forged fingerprints, obtaining full control of PA is required. From a worldwide point of view, the risk is low due to the fact that TA is optional. From a European point of view, the risk is medium because TA is obligatory in the EU and it implies a potential privacy issue for EU passports.

Reference (OID)	**Signature algorithm**	**Digest algorithm**
id-TA-RSA-PSS-SHA-256	RSASSA-PSS	SHA-256
id-TA-RSA-PSS-SHA-512	RSASSA-PSS	SHA-512

Table 6: Terminal authentication with RSA specifications [ICA21c].

Reference (OID)	**Signature algorithm**	**Digest algorithm**
id-TA-ECDSA-SHA-224	ECDSA	SHA-224
id-TA-ECDSA-SHA-256	ECDSA	SHA-256
id-TA-ECDSA-SHA-384	ECDSA	SHA-384
id-TA-ECDSA-SHA-512	ECDSA	SHA-512

Table 7: Terminal authentication with ECDSA specifications [ICA21c].

4. Research

The previous section elaborated on the security protocols and all risks involved in regard to quantum computing. The results of this analysis have been summarized in Table 8 and in Table 9 a more detailed overview is given of the involved cryptosystems for each security mechanism. In Table 8 we see that passive authentication has a high impact on security and the scope is worldwide and in Table 9 we see the algorithms used within passive authentication, which are all classical algorithms which are vulnerable for quantum attacks in the future. It is clear that passive authentication is the most vulnerable protocol within the eMRTD in regard to the post-quantum threats, for Europe this extends to EAC. PA is not only the most vulnerable protocol, it is also the most important protocol for the security of travel documents. Due to the passiveness of the protocol, it is possible to create offline attacks to obtain the private keys of the CSCA or the corresponding document signer. Until today it is still uncertain when a quantum computer will arrive exactly with enough computational power to perform successful attacks on asymmetric cryptosystems. Experts estimate that a capable quantum computer will become available within the next two decades [MCJ+16, Mos18]. One could say that there is still enough time to propose a post-quantum solution for electronic travel documents. However, solely proposing a solution is not sufficient, because such a solution also needs to be standardized and adopted throughout the world. With the validity of travel documents exceeding a decade, it is possible to have an overlapping era where passports using a classical implementation of passive authentication can be broken.

Protocol	**Vulnerable ciphers**	**Impact eMRTD Security**	**Scope**
BAC	3DES	Low	Worldwide
PACE	3DES	Low	Worldwide
PA	RSA, DSA, ECDSA	High	Worldwide
AA	RSA, ECDSA	Low	European Union
CA	DH, ECDH	Low	European Union
TA	RSA, ECDSA	Medium	European Union

Table 8: Preliminary impact assessment eMRTD security

In this research, our goal is to establish a proof of concept for quantum resistant passive authentication. In order to do so, we first require to set the scope of our research, which will be discussed later on in this section. Since post-quantum algorithms differ in performance we need to cherry pick the algorithm or algorithms that best suit our use case. This means we need to evaluate the algorithms being considered within this research and argue which of these algorithms would fit the best in our research. When we have selected a suitable algorithm, we need to create a public key infrastructure for passive authentication using the post-quantum algorithms. This will be done by investigating the open source crypto-library Bouncy Castle and setup a PKI using the Bouncy Castle post-quantum algorithm implementations. The last step is creating an eMRTD which resembles the ones currently in the field. As final step to validate if our implementation is able to function in the field, we use the existing JMRTD framework. By the end of this research we have covered all parts of the life cycle of the eMRTD in order to make the eMRTD quantum resistant.

Protocol	**Purpose**	**Cryptosystems**	**Quantum threats**
BAC	Authentication	3DES	—
PACE	Authentication & secure communication	3DES, AES, DH, ECDH	DH, ECDH
PA	Document integrity	RSA, DSA, ECDSA	RSA, DSA, ECDSA
AA	Clone prevention	RSA, ECDSA	RSA, ECDSA
CA	Clone prevention & secure communication	3DES, AES, DH, ECDH	DH, ECDH
TA	Access control	RSA, ECDSA	RSA, ECDSA

Table 9: Overall overview of the eMRTD security protocols including the cryptosystems and quantum computing threats to those cryptosystems.

Table 10 shows the quantum algorithms that are being investigated by the NIST. These algorithms have been thoroughly reviewed and have advanced to the fourth round or have already been selected for standardization. In order to establish a quantum proof solution for passive authentication, one or more algorithms process must be compatible with the PKI used in passive authentication. For passive authentication, digital signature algorithms are the basis of establishing a PKI. Table 10 shows exactly three signature algorithms fit for that purpose: CRYSTALS-Dilithium, FALCON and SPHINCS+. We picked the NIST signature algorithms that are being standardized, because they have survived four rounds of attacks and evaluation. In earlier rounds of the NIST standardization process, there were more potential standardization candidates, but they have been dropped in favor of these digital signature algorithms [AAC+22]. To widen the diversity of digital signature algorithms, as the two of the thee signature algorithms are lattice based, the NIST has opened another call for proposals in 2022 [CML22].

4.1. NIST STANDARDIZATION CANDIDATES

There are three cryptosystems that are being standardized by the NIST as a result of the third round post-quantum cryptosystem evaluation [AAC+22]. Since they are being standardized by the NIST, they are suitable candidates for this research. We will

Name	Type	Purpose	Status
CRYSTALS-Kyber	LWE-based	Public-Key Encryption/KEMs	To be standardized
BIKE	QC-MDPC	Public-Key Encryption/KEMs	Advanced to fourth round
Classic McEliece	Code-based	Public-Key Encryption/KEMs	Advanced to fourth round
HQC	QC-MDPC	Public-Key Encryption/KEMs	Advanced to fourth round
SIKE	SIDH	Public-Key Encryption/KEMs	Advanced to fourth round
CRYSTALS-Dilithium	Lattice-based	Digital Signatures	To be standardized
FALCON	Lattice-based	Digital Signatures	To be standardized
SPHINCS+	Hash-based	Digital Signatures	To be standardized

Table 10: NIST fourth round standardization candidates for post-quantum cryptography [AAC+22].

provide a short overview what the properties of these cryptosystems are and why they are considered for this research. For the mathematical background and the detailed technical implementation we refer to the corresponding papers and RFCs.

CRYSTALS-Dilithium is a lattice-based digital signature scheme built upon the hardness of finding short vectors in lattices [BDK+21]. According to the benchmark statistics in [BDK+21], the signature size of the configuration for NIST level 5 is 4595 bytes and the public key size is 2595 bytes. The study of ENISA considers the keys and signatures to be medium-sized [BHL22].

FALCON is a lattice-based digital signature algorithm and is an acronym which stands for "Fast Fourier lattice-based compact signatures over NTRU" [FHK+20]. Falcon comes in two flavors, Falcon-512 and Falcon-1024, where Falcon-512 has a security level of 1 and Falcon-1024 has a security level of 5. Falcon1024 also has a medium-sized public key and signature according to ENISA [BHL22]. The performance characteristics show this in [FHK+20] for Falcon-1024, where the public key size is 1793 bytes and the signature size is 1280 bytes in the test results.

SPHINCS+ is a stateless hash-based signature framework [BHK+19] and continues on the ideas of SPHINCS [BHH+15]. The hardness of SPHINCS+ depends on the security properties of hash functions [BHK+19]. The hash functions currently supported for this scheme are SHAKE-256, SHA-256 and HARAKA [ABB+22]. SPHINCS+ is one of the hash-based signature schemes that is stateless, which means that there is no state of the private key to maintain to prevent jeopardizing the security of the scheme. One of the properties of SPHINCS+ is that the size of the private and public key pairs are small; the SPHINCS+ configuration SPHINCS+-256f for example has a public key of 64 bytes and a private key of 128 bytes [BHK+19]. However the signature size in this case almost reaches 50 KB [BHK+19].

4.2. Stateful hash-based signature schemes

In addition to the NIST standardization candidates, there are also two stateful cryptosystems that are interesting in the scope of this research. This section will provide a global overview of the two stateful hash-based signature schemes LMS and XMSS and the possibilities for the context of this research. XMSS and LMS are similar schemes both consisting of two components, an One-Time Signature Scheme (OTS) scheme and a method to create a public key which has a long lifetime [CAD$^+$20]. XMSS [BDH11] and LMS [LM95, MCF19] use their own adaption of the Winternitz signature scheme as OTS [CAD$^+$20] and use single and multilevel Merkle trees [Mer79, Mer90] as method to construct these large keys. Stateful hash-based signature schemes are not considered for general use according to the NIST since they require careful state management of the private keys [CAD$^+$20]. Any accidental reuse of an OTS private key leads to a security collapse of the scheme [CAD$^+$20]. The NIST mentions three requirements for practical use of hash-based signature schemes [CAD$^+$20]:

1. The application requires a secure digital signature scheme in the near future.
2. The implementation has a long lifetime.
3. The signature scheme will not change once it has been implemented in the application.

Although these requirements are relevant for passive authentication, there are some important challenges to include XMSS and LMS in our research. The problem regarding the scope of this research is that XMSS and LMS have a wide range of parameters in their implementations [HBG$^+$18, MCF19] and each parameter considers a trade-off between security and performance [KF17]. If XMSS and LMS would be considered in this research, this would require a thorough research to determine the best set of parameters to obtain a high level of security and keys that would fit on a chip for specifically passive authentication. This would increase the time required to perform the entire research considerably. Also a study has to be performed to determine the proper and safe implementation of XMSS and LMS to be able to used in practice. Therefore, the study on stateful hash-based signature schemes will be considered out of scope for this research but remains interesting for future research.

As a brief overview, stateful hash-based signature schemes have the following characteristics:

- The key size of stateful hash-based signature schemes is considered small and signatures can be generated fast. However, this depends on the chosen parameters of the security schemes.

- The size of the signatures is large and the key generation times is considered slow for stateful hash-based signature schemes.
- Difficult for engineering implementations, because an environment is required where the state of the private key is managed in such a way that a single key can **never** be used more than once.

ENISA mentions that the performance of XMSS and LMS, especially the signature size, is similar to lattice based signature schemes [BHL22]. This however depends on the configuration used for the stateful hash-based signature algorithm, but when compared to stateless hash-based signature algorithms such as SPHINCS+, the stateless algorithms have a larger signature than the stateful algorithms [KF17].

4.3. Scope

There are several post-quantum algorithms that could be used during this research. The scope is limited to the algorithms that are being considered for standardization by the NIST and have been admitted to the fourth standardization round. Also, for passive authentication only algorithms designed for digital signatures will be considered. The signing algorithms being investigated are *CRYSTALS-Dilithium (shortened as Dilithium)*, *FALCON* and *SPHINCS+* as mentioned in Table 10. These algorithms are free to use without any patent restrictions or usage fees.

Since this research involves algorithms which are still being tested, attacked and evaluated, it is important to keep track of any research involving one of the considered post-quantum algorithms. If one of the candidates considered would be rendered insecure due to ongoing research, the next algorithm listed in the comparison will be chosen as candidate instead. If this happens in the course of the next research questions, it is key to compose a PKI structure that allows interchanging the underlying crypto primitives in a modular way.

Since this is an engineering research, the post-quantum algorithms themselves will not be mathematically validated within this paper, as such process is beyond the scope for this research. The algorithms will be considered building blocks in order to be able to propose a quantum proof implementation of eMRTDs. The scope of this research is to evaluate and use the implementation of these algorithms by using the Bouncy Castle library in order to construct a quantum proof PKI specifically to be used for eMRTDs.

The organizational and political concerns are also not considered in the scope of this paper, because the implementation of passive authentication is different for each country worldwide. This would require another research approach to determine the organizational challenges to standardize the embedding of post-quantum algorithms within the scope of passive authentication.

In the preliminary research it is clearly visible that passive authentication for electronic travel documents is at risk caused by the increasing threat of quantum computing. Even though a fully operational and capable quantum computer is still assumed decades away, it is still feasible to propose an initial solution to keep our borders safe in the future. Therefore, this research focuses on the question how to create a quantum-proof implementation of passive authentication for electronic machine readable travel documents. The goal is to provide a proof of concept implementation for passive authentication using PQC algorithms.

5. Best suited algorithm for passive authentication

Before a proof of concept can be established, a benchmark, comparing execution efficiency and storage space, is required in order to determine the algorithm best suited for passive authentication for eMRTDs. This is performed using the Bouncy Castle library, since Bouncy Castle version 1.76 supports the considered algorithms. Each post-quantum algorithm has several implementations and configurations which are considered during this research. Before continuing on the benchmarking, some elaboration is required on the configurations which are being considered during this research.

For the lattice-based signature algorithms such as Dilithium and Falcon, the configurations as shown in Table 21 of the Appendix are being considered. This includes the round 2 introduced variants of Dilithium in which SHAKE is replaced by AES for efficiency purposes for specifically the expansion of the matrix [BDK+21]. For the lattice based signature algorithms, there are a total of 8 configurations considered. Note that the OIDs mentioned are the OIDs which are referred to in the Bouncy Castle library and may be subject to change in future versions.

SPHINCS+ on the other hand has a large number of configurations available. An exhaustive list of each configuration for SPHINCS+ is shown in Table 22 of the Appendix. There are a total of 36 unique instantiations of the SPHINCS+ algorithm which can be identified by three characteristics. The first one is the type of hash algorithm used within the configuration, there are three hash algorithms which can be used within the configuration: SHA2, SHAKE or HARAKA. The next characteristic is the difference between the simple and the robust variant. Since round 2, SPHINCS+ has introduced the tweakable hash functions and SPHINCS+ has made the separation between the already established instantations (robust) and the new implementations (simple) which are faster but omit a security argument [ABB+22]. The third characteristic is the f- and s-variant for each instantiation, where the f-variant is speed-optimized and the s-variant is size-optimized [ABB+22]. It would make sense to include only the s-variants for the research, but for completeness, all variants of SPHINCS+ are included. Table 22 also mentions the NIST security level which is used to measure the cryptographic strength of the post-quantum algorithms with the current NIST standards in symmetrical cryptography. In total, there are five levels which the NIST used during the evaluation of the PQC standardization candidates which are based on criteria as shown in Table 11. In the result overviews, the configurations are grouped by their security level. This will yield a better overview for considering the best suited algorithm for passive authentication purposes. In total we evaluate 44 configurations within our research.

In order to select the best suited algorithm for passive authentication, it is necessary to select unique properties of the listed post-quantum signing algorithms. We con-

Level	Security description
I	Algorithm is at least as hard to break as AES128 (exhaustive key search)
II	Algorithm is at least as hard to break as SHA256 (collision search)
III	Algorithm is at least as hard to break as AES192 (exhaustive key search)
IV	Algorithm is at least as hard to break as SHA384 (collision search)
V	Algorithm is at least as hard to break as AES256 (exhaustive key search)

Table 11: NIST security levels used for determining the cryptographic strength of a post-quantum algorithm [Moo18].

sider six properties for this research and these properties are listed in Table 12. These properties can be divided into two categories. The first category is performance-related, such as the signature validation time, signature generation time and the time needed to generate a key pair. The other category is storage-related, which include the size of the public key, the size of the private key and the size of the generated signature by the algorithm. In Table 12, the variables which are important for passive authentication, are listed in bold.

Variable	Category
Private key size	Storage
Public key size	Storage
Signature size	Storage
Key Generation time	Performance
Signature generation time	Performance
Signature validation time	Performance

Table 12: Properties investigated during this research to determine the best suited PQC SA for Passive Authentication.

For practical implementation, the properties signature size, public key size and signature validation time are the most important for passive authentication, since the public key and signature are stored on the chip. The signature validation time is of importance at the border to determine the authenticity of a travel document. The other properties are important during the production of the travel documents and are considered less important in regard to passive authentication in this research.

5.1. Benchmarking within the Java JVM

The PQC crypto systems are all benchmarked using low level programming languages such as C [FHK+20, BDK+21, BHH+15]. Since the implementation of the PQC PKI will be done in Java, a language on a higher abstraction level, it is necessary to evaluate the impact of the Java Virtual Machine on the performance of the considered algorithms. However benchmarking on a higher abstraction level is more complicated due to the fact that the compiled code is not being run directly at the hardware level. This means that benchmarking performance in Java is not straightforward. An approach similar to the C benchmarks is not feasible here, because the JVM (Java Virtual Machine) and JIT (Just In Time compiling) are using optimization techniques to speed up execution times [CBL+21]. Examples of these optimizations are dead code elimination, loop optimization and constant folding [CBL+21]. To counter and suppress these side effects of the JVM, this research includes the use of the OpenJDK Java Microbenchmarking Harness (JMH) framework [Shi13]. This framework allows us to create micro-benchmarks which are needed to determine the best candidate for the passive authentication proof of concept. Even though the JMH Framework is a useful tool for creating micro-benchmarks, it still remains easy to make mistakes or to establish bad practices when benchmarking in Java as shown in a recent study in regard to JMH [CBL+21]. It is necessary to carefully consider the risks present in the JMH framework and to verify that there are no fundamental mistakes in the benchmarking. To validate the benchmarks on any of these mistakes, the Spotbugs plugin as described in [CBL+21] will be used.

For the purpose of determining the best suited signature algorithm for passive authentication, we have composed six benchmarks within Java and are divided into two projects [6]. One project is responsible for measuring the storage-related properties and the other project uses JMH for measuring the performance-related properties. Since JMH has a delicate and elaborate framework, the benchmarks for the storage-related properties are separated into another Java project to prevent unintended interference of or onto the JMH framework. These projects are the basis for benchmarking the PQC signature algorithms in Java. All benchmarks are performed on a freshly installed Ubuntu 22.04.3 (LTS, physical desktop) system with an Intel Core i5-7500T @ 2,75 GHz with 4 GB of RAM. Specifically for the used setup, processor performance optimization techniques, such as Intel's Turbo Boost, Intel Speedstep and C-states are disabled to prevent potential interference and inconsistent results during the benchmarking process.

Measuring performance: The benchmarking of the performance-related properties is fairly straightforward. Each benchmark within this project is divided into a preparation or setup phase and an execution phase. This also applies to the implementation using JMH and the underlying JMH framework ensures that the proper steps are taken based on the supplied configuration. The benchmark results in JMH are measured in microseconds (μs) and in each benchmark the average time of execution of

[6]The two used project created specifically for this research are listed in the Appendix in Table 27 under PQC-BC-Benchmark and PQC-BC-Benchmark-Keysizes.

the measured cryptographic function is measured. The steps executed in the setup phase are not part of the benchmark result. JMH executes the method as many times as possible within 20 seconds and measures the average time it took to execute the body of the benchmark function. For this research, a JMH benchmark configuration with 100 iterations and 2 warm-ups is used for all considered algorithms. Afterwards we extract the results stored in the JMH output files using a Python script.

Measuring storage cost: Measuring the storage-related properties cannot be measured using JMH, and in order to keep the benchmarks separated, we have created another Java project and a simple framework to measure the storage-related properties of the different PQC algorithms using the Bouncy Castle library. Each benchmark measuring the storage-related properties will be executed 100 times and the results are stored in CSV format on disk and will be extracted by using a Python script. Since the storage values are expected to be static, the framework is simple and straightforward. In order to measure the signature size, a fixed size has been set for the input of the signature generation function. As input value to generate the signature, a 32 bytes initialized array with random values is used which is hashed and signed by the specified PQC algorithm. To establish a bridge between the algorithm benchmarks and a practical implementation, this randomized array of 32 bytes is used to benchmark the signature generation time, validation time and the size of the signature itself. 32 bytes (256 bits) is exactly the size of the hashing algorithm with the largest fixed-length in our PQC algorithm list and is representative of determining the best suited post-quantum algorithm to be used for passive authentication. In Bouncy Castle, the private keys are stored in PKCS#8 format. This leads to larger files when compared to the algorithm's documentation due to the encoding standards. This also applies for the public key size, because these are stored in X.509 format. Since this research is focused on the practical applications of the algorithms, these deviations are not considered harmful.

5.2. ALGORITHM BENCHMARK RESULTS

The benchmarks of the storage-related properties (Table 13) and the performance-related properties (Table 14) are briefly summarized. For the sake of brevity, the extended results for all algorithms and variants are shown in the Appendix in Tables 23 and 24 respectively. For SPHINCS+ we have selected the SHA2-variant to be shown in the abbreviated table, as this performs better overall compared to the SHAKE and HARAKA variants.

The first observation is that the SPHINCS+ signature size is large for all considered variants compared to the lattice-based variants. It is even the case that the 192f and 256s/f variants are too large to be used in the CSCA to sign a Document Signer, because the signature in the document signer will exceed the 32k limit on chip of the passport. Another observation is that the private key of the SPHINCS+ algorithms is small compared to Dilithium and Falcon. This also applies for the public key, however the effect of this benefit is cancelled out when combined with the size of the signature, which will both be stored in a certificate.

Algorithm	**Private Key**	**Public Key**	**Signature**	**Sec. Level**
Falcon-512	2223	915	* 655	I
SPHINCS+ (128f)	101	58	17088	I
SPHINCS+ (128s)	101	58	7856	I
Dilithium2	3902	1336	2420	II
Dilithium3	6014	1976	3293	III
SPHINCS+ (192f)	134	74	35664	III
SPHINCS+ (192s)	134	74	16224	III
Dilithium5	7518	2616	4595	V
Falcon-1024	4143	1811	* 1271	V
SPHINCS+ (256f)	168	90	49856	V
SPHINCS+ (256s)	168	90	29792	V

Table 13: Storage-related benchmark results of PQC algorithms grouped per security level. Results of the measurements are in bytes. (*) Falcon has a varying signature length. The listed result is the average size over 100 iterations. Elaborated results of all algorithms individually can be found in the Appendix (Table 23).

Further on, lattice based algorithms Dilithium and Falcon are performing well in terms of signature generation compared to SPHINCS+. The speed differs when using different implementations of SPHINCS+. The SPHINCS+ 256s variants range from 227 milliseconds up to 10 seconds to generate a signature on a 256 bit message corresponding to a message digest, this is slower when compared to Falcon-1024 and Dilithium5 in the same security level group. For passive authentication on the border this does not matter, since the signature for the Document Signer is only generated during the production of the passport. However for the passport production industry, this could slow down the production process. In contrast to the signature generation time, the time required to validate the signature using any of the benchmarked algorithms is low for all PQC candidates, the SPHINCS+ algorithms are generally slower compared to Dilithium and Falcon. However, the slowest time measured (HARAKA-256f-robust) is 26ms, which is still quite fast for validation and will not cause any problems for the validation part of the passive authentication process. Interesting to mention is that the expectation was that the signature size would be consistent and static for all considered configurations, however for one algorithm this is not the case: Falcon. During the benchmark, the results showed that the signature size of Falcon is different per signature using the same input. Normally, signature algorithms have a fixed signature length for the same input, but as described in the specifications, this does not apply for Falcon (see 3.11.6 in the Falcon paper) [FHK+20].

Algorithm	**Key generation**	**Sign. generation**	**Sign. validation**	**Sec. Level**
Falcon-512	15396	1323	74	I
SPHINCS+ SHA2-128f-robust	5815	140844	8545	
SPHINCS+ SHA2-128s-robust	373859	3150242	2911	
SPHINCS+ SHA2-128f-simple	2789	67842	3900	
SPHINCS+ SHA2-128s-simple	174893	1525860	1357	
Dilithium2	158	563	170	II
Dilithium2 (AES)	290	737	279	
Dilithium3	277	1011	264	III
Dilithium3 (AES)	529	1278	538	
SPHINCS+ SHA2-192f-robust	8442	225112	12277	
SPHINCS+ SHA2-192s-robust	548427	5406656	4372	
SPHINCS+ SHA2-192f-simple	4016	108477	5653	
SPHINCS+ SHA2-192s-simple	260883	2706109	2023	
Dilithium5	428	1196	436	V
Dilithium5 (AES)	859	1693	862	
Falcon-1024	42182	2745	148	
SPHINCS+ SHA2-256f-robust	29642	605148	16539	
SPHINCS+ SHA2-256s-robust	479833	5885411	8296	
SPHINCS+ SHA2-256f-simple	10605	227719	5796	
SPHINCS+ SHA2-256s-simple	173780	2357170	2997	

Table 14: Performance-related benchmark results of PQC algorithms using the JMH framework in Java and are grouped per security level. Results are the average time needed to execute the corresponding Bouncy Castle function (in μs). Elaborated results of all algorithms individually can be found in the Appendix (Table 24).

To summarize, concerning the properties which are important for passive authentication, only the size of the signature poses difficulties to passive authentication. Specifically the SPHINCS+ 192f and 256s/f variants are not suitable for passive authentication due to their large signature size. Comparing the storage size to Dilithium and Falcon, SPHINCS+ is simply outmatched for the task. Even though the public key size is the smallest for SPHINCS+ compared to Dilithium and Falcon, the sum of the two will simply cancel out the benefits of the small public key size. So Dilithium and Falcon are the preferred candidates for passive authentication, where Falcon scores the best for having the smallest signature size and the fastest validation time.

6. Post-quantum PKI for passive authentication

In the previous section, the results showed that the lattice based signature algorithms Dilithium and Falcon are performing well. The next step is to extend these results into the world of passive authentication. In order to do so, the benchmarks in the previous section are extended by introducing the X.509 aspects which passive authentication relies on. The following properties will be evaluated in the second benchmark as shown in Table 15. The properties that matter the most for passive authentication in practice are the size of the certificate, which is being stored on the chip, and the time required to validate the chain of trust.

Variable	**Category**
Certificate size	Storage
DS CSR generation time	Performance
Certificate signing time	Performance
Certificate validation time	Performance

Table 15: Properties of the X.509 certificates which will be evaluated in this research for passive authentication.

6.1. Passive authentication X.509 certificate requirements

Before being able to benchmark the algorithms in practice, it is required to investigate the current standards regarding the PKI for passive authentication and integrate them within the benchmarks. ICAO 9303 (part 12) defines the Public Key Infrastructure for eMRTDs and lists all required and optional properties of the X.509 certificates used within passive authentication [ICA21d]. These specifications differ for each of the components within PKI structure of passive authentication. In this investigation the scope is limited to the CSCA and the Document Signer components only, since the other components are also listed within the standardisation documentation. In the current situation where ICAO 9303 (version 8) applies, only RSA, DSA and ECDSA are the only allowed signature algorithms for passive authentication. For the hash algorithms, only the algorithms SHA-224, SHA-256, SHA-384 and SHA-512 are permitted to be used. These cryptosystem requirements are ignored for the implementation of the post-quantum algorithms which are being used within the CSCA and the Document Signer certificates during this research, because otherwise it is impossible to meet these requirements when the cryptosystems are replaced by a post-quantum algorithm in this research. In this research, a CSCA and Document Signer have been constructed using the investigated post-quantum algorithms in combination with all required components and extensions alongside their corresponding criticality using the Bouncy Castle library in Java. The ICAO 9303 (part 12) [ICA21d] is used as a reference document for this project. For all detailed requirements we refer to this document, since it is an extensive list of required certificate components and extensions which are too exhaustive to mention within this paper.

6.2. Benchmarking certificate properties

The benchmarks on the properties for specifically the certificates are performed using the same methodology as for the benchmarks earlier in this paper. The previous setup is reused and new benchmarks are added which focus on the application of the algorithms within X.509 certificates. The results of the benchmark for specifically the certificate size (storage) are shown in Table 16. The results of the performance-based properties such as the time required to generate a CSR, a self-signed (CSCA) certificate and to validate the certificate chain are shown in Table 26.

Algorithm	**Certificate Size**	**Security Level**
Falcon-512	2182	I
SPHINCS+ (128f)	17768	
SPHINCS+ (128s)	8536	
Dilithium2	4380	II
Dilithium3	5893	III
SPHINCS+ (192f)	36360	
SPHINCS+ (192s)	16920	
Dilithium5	7835	V
Falcon-1024	3694	
SPHINCS+ (256f)	50568	
SPHINCS+ (256s)	30504	

Table 16: Results of the benchmark on the storage properties of the X.509 certificates using the Bouncy Castle library (in bytes). The extended version can be found in the appendix for all individual variants in Table 25.

In Table 16 it is clearly visible that the variants of the SPHINCS+ using the 192f, 256f and 256s configuration are impossible to be used as a CSCA for the document signer. The signature of these signature algorithms will cause the Document Signer certificate to be too large to be practically be stored on the chip. The difference between the hash-based signature algorithms and the lattice-based signature algorithms is also very clear in terms of certificate size within this table. Falcon-1024 yields a certificate with a size of 4kB, whereas the SPHINCS+ 256f variant is exceeding 50kB. Dilithium5 on this configuration reaches 8kB in size, but this is still significantly smaller than the hash-based variants on the same security level.

There are also differences in the performance of the algorithms when used in a practical context. For the sake of clarity, the results for the benchmarks regarding the performance properties can be seen in the Appendix in Table 26. The validation of the full certificate chain is not very interesting to analyse further, since the slowest validation of the certificate chain took 26ms, which is still fast in practice. Where it gets more interesting is the performance difference between the SPHINCS+ configurations. In Table 17 we have extracted the security level 5 results and it is clearly visible that the lattice-based signature algorithms are outperforming the hash-based variants. The f-variant (fast) is notably faster than its s-variant (size), but the size of the certificate generated by this variant would be over 50kB. If the s-variant is picked however, it would take almost 6 seconds to generate and sign a certificate using the robust variant and over 2 seconds using the simple variant. Which is quite long for generating a certificate compared to the other algorithms. Objectively observing the results overall, SPHINCS+ is not a feasible candidate for passive authentication. Falcon scores very well for generating very small signatures and is followed by Dilithium. Dilithium performs the best results when considering the performance-related properties.

Algorithm	**Generate CSR**	**Create CSCA**	**Validation**
Dilithium5	1145	1291	466
Dilithium5 (AES)	1659	1712	897
Falcon-1024	2773	2764	184
SPHINCS+ SHA2-256f-robust	628469	618112	16944
SPHINCS+ SHA2-256s-robust	5917595	5903589	8445
SPHINCS+ SHA2-256f-simple	236426	235911	6044
SPHINCS+ SHA2-256s-simple	2351470	2358679	3027

Table 17: Parts of the results of the benchmark for the performance properties of the X.509 certificates using the Bouncy Castle library (in μs). Full overview of the results can be seen in the Appendix in Table 26).

6.3. Algorithm selection

In the previous results it is clear that Falcon and Dilithium are the preferred candidates for passive authentication. However, there are different security levels available for these algorithms. It is important to carefully pick an algorithm variant and argument why this variant is future proof. The CSCA is the root of the PKI for passive authentication and needs to be secure for at least ten years. The easy solution is to pick the highest level algorithm variant available, but the question is if this is absolutely necessary. In order to view the current landscape, we have investigated three Masterlists publicly available in the world. A Masterlist is a way of exchanging CSCA certificates easily and securely between countries. In these Masterlists, it is possible to investigate which algorithms are used and what the cryptographic strength is for the used algorithms. The overview of the results are listed in Table 18 and in this overview it is clearly visible more than 60 percent of the CSCAs uses RSA-4096.

PQC algorithm	Dutch	German	ICAO
ECC-256	28	28	24
ECC-384	39	39	34
ECC-512	6	6	6
ECC-521	4	5	5
RSA-2048	1	1	5
RSA-3072	35	37	26
RSA-4096	218	220	178
RSA-6144	5	5	7

Table 18: Masterlist analysis to view the CSCA algorithm usage throughout the world.
Dutch, Nov. 2023: `https://www.npkd.nl/masterlist.html`,
German, Feb. 2024: `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ElekAusweise/CSCA/GermanMasterList.html`,
ICAO, Dec. 2023: `https://www.icao.int/Security/FAL/PKD/Pages/ICAO-Master-List.aspx`
See Table 27 - Masterlist Tool repository for the actual used Masterlist files.

In order to compare the classical and the post-quantum algorithms in terms of security strength, it is necessary to have an overview which describes all different algorithms and cryptographic strength in ascending order. This is done by investigating the security strength of the post-quantum algorithms, which can be seen in Table 11 mentioned earlier, and by investigating the cryptographic strength of classical algorithms in the NIST documentation related to key management in general [Bar20]. The post-quantum algorithms also have a classical security analysis, as they still need to be secure in the classical non-quantum world. As a result, this yields the following overview as mentioned in Table 19.

RSA-4096 is currently widely used within CSCA-certificates, however it is not listed explicitly in the advisory as mentioned in Table 19. It is reasonable to assume through simple interpolation that RSA-4096 is positioned in the category 128-192 bits security. It is clear to see that the level 3 post-quantum algorithms are of a higher level of security. The key of the CSCA is valid for 10 to 15 years, so also in a post-quantum era a high level signature algorithm is required. This would mean that for the CSCA certificate, a post-quantum algorithm of level 3 or higher is needed. A conservative approach will be used for our use case, which means that the lattice based algorithms from the 5th category (Dilithium5 and Falcon-1024) will be used in the proof of concept for the CSCA. For the Document Signer, there is no publicly available information available. The Dutch document signers use RSA-2048 and according to Table 19 this yields 112 bits of classical security. In a PQC era this would mean that the document signer must use a level 1 or 2 PQC algorithm. So specifically for this use case, Falcon-512 and Dilithium2 will be used for the document signer. This base line is not

PQC Level	Algorithm	Configuration	Classical bits
—	RSA	1024	< 80
—	DSA	1024	< 80
—	ECDSA	160-223	< 80
—	RSA	2048	112
—	DSA	2048	112
—	ECDSA	224-255	112
—	RSA	3072	128
—	DSA	3072	128
—	ECDSA	256-383	128
I	Falcon	512	128
I	SPHINCS+	128 (all)	128
II	Dilithium	2	128-192
—	RSA	7680	192
—	DSA	7680	192
—	ECDSA	384-511	192
III	Dilithium	3	192
III	SPHINCS+	192 (all)	192
—	RSA	15360	256
—	DSA	15360	256
—	ECDSA	512+	256
V	Dilithium	5	256
V	Falcon	1024	256
V	SPHINCS+	256 (all)	256

Table 19: Overview of classical and post-quantum algorithms, ordered by security level in order to determine a suitable candidate for the document signer and the CSCA.

set in stone, so the proof of concept will be created in such a way that it allows to easily change the used algorithms for passive authentication.

7. POST-QUANTUM CHIP SIGNING AND VERIFICATION

In the previous section, we have created a PQC PKI structure for passive authentication. This PKI will be used to construct the chip and sign the corresponding data groups with the selected post-quantum algorithms. We have created a Java application which is responsible for the creation process of the eMRTD. This application creates, signs and writes the data to the chip which is used in this research. Dilithium and Falcon will be used to create a proof of concept quantum resistant eMRTD for passive authentication, since these are well suited for the process as investigated earlier.

As mentioned in the introduction, the LDS of the eMRTD is defined according to the ICAO 9303 (part 10) [ICA21b]. The goal of the research is to create a practical proof of concept which resembles a real eMRTD. In order to not deviate from the scope of this research, only the required components of the eMRTD will be considered in the proof of concept. The proof of concept consists of data group 1 (MRZ) and 2 (Photo), the EF.SOD and the EF.COM. Since EAC is not in the scope of this research and adds more complexity to the proof of concept, these elements are out of scope. Figure 9 shows the elements which are being included and will elements will be omitted in this proof of concept. Even though BAC is deprecated, implementing PACE on the smartcard used in practice required a significant amount of work and does not contribute to the research objectives. Therefore, the proof of concept currently only supports BAC.

The Document Security Object (EF.SOD) is the most important component for passive authentication, because the hashes of the data groups, the signature over these hashes and the document signer are stored here. In short, the EF.SOD is a custom ICAO object (LDSSecurityObject) and is signed by using Cryptographic Message Syntax (CMS) as specified in RFC 3369 [Hou02]. CMS is a protocol to digest, sign, authenticate or to encrypt any type of data [Hou02]. CMS is structured in such a way that it does not matter which digest algorithm or signature algorithm is used. This means that only the algorithms and their corresponding cryptographic functions need to be defined in the software. The CMS functionality is present in Bouncy Castle and can be used to construct the quantum resistant signature provided by the Document Signer for passive authentication. From a bird's eye view, the structure of the EF.SOD is shown in Figure 10. We use the EF.SOD V1 implementation for our proof of concept as defined in the ICAO 9303 (part 10), as the V0 variant is considered legacy and is deprecated [ICA21b].

Our implementation uses the JCOP J3E082-EXS card with the eMRTD application embedded in the card. Communication with the chip is performed by using Application Protocol Data Unit (APDU) commands according to the standards of the ISO 7816-4 [ISO20]. The widely used GlobalPlatform specifications are used for the man-

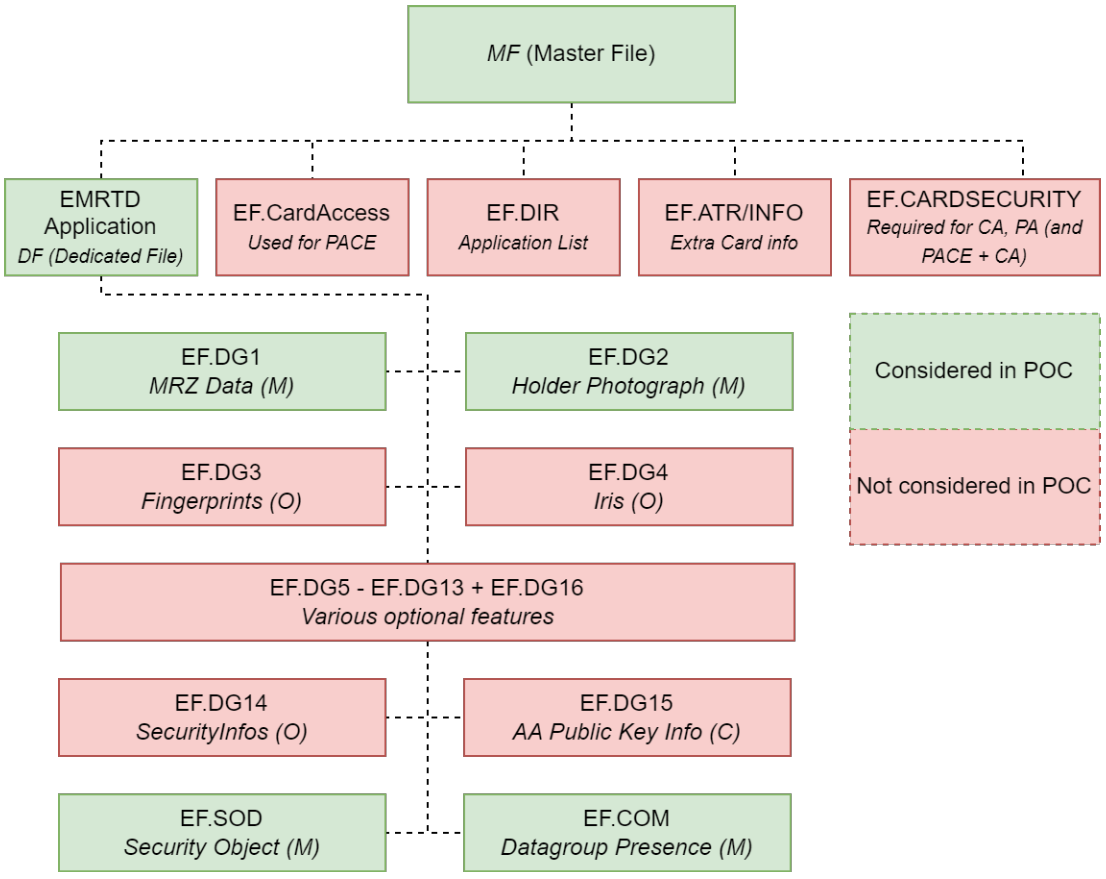
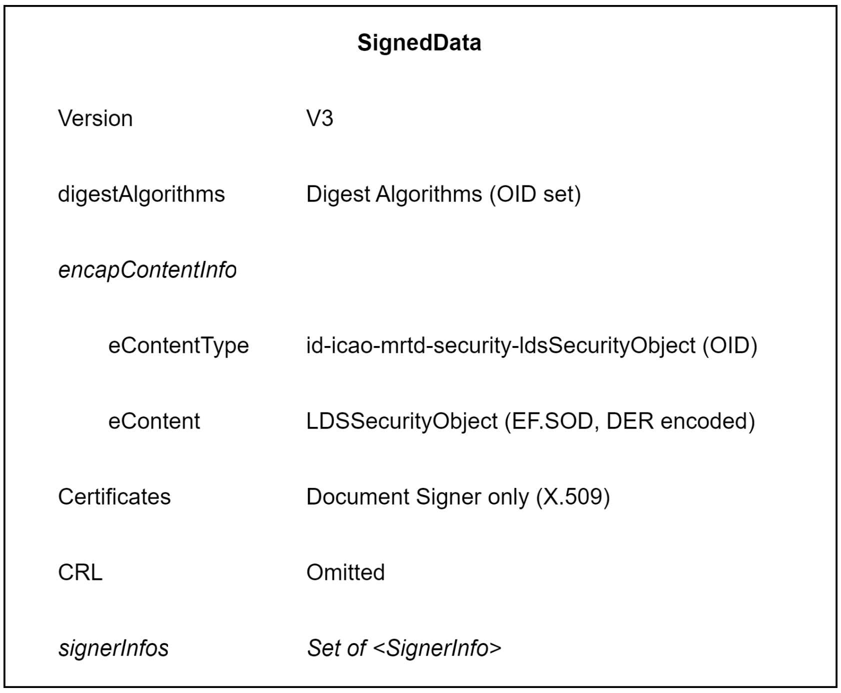


Figure 9: Structure of the LDS considered within the proof of concept for PA.

agement of the applets on the eMRTD chip [Glo18], and particularly for the chip writing implementation in Java, we use the open source project GlobalPlatformPro[7] as supporting library. In addition to GlobalPlatformPro, we use the open source JMRTD project for composing the data structures for the eMRTD, such as the MRZ (DG1), the encoding rules of the photo (DG2) and the other components as shown in Figure 9. For the personalization of the card, we use the fictional identity of 'Vestram Identitatem' (Your Identity) including a vector generated image for this identity. The main reason for this is to prevent using personal data of copyrighted material in this research. For the EF.SOD implementation it was necessary to work around the JMRTD library for establishing the CMS signature, because JMRTD hard coded the necessary components for creating and signing the EF.SOD, such as the digest algorithms used in the post-quantum signature algorithms. We did not adapt the JMRTD library, as this research focuses on the Bouncy Castle library in particular. Bouncy Castle contains the necessary components in order to construct and sign the EF.SOD and we

[7]https://github.com/martinpaljak/GlobalPlatformPro



SignedData

Version	V3
digestAlgorithms	Digest Algorithms (OID set)
encapContentInfo	
eContentType	id-icao-mrtd-security-ldsSecurityObject (OID)
eContent	LDSSecurityObject (EF.SOD, DER encoded)
Certificates	Document Signer only (X.509)
CRL	Omitted
signerInfos	*Set of <SignerInfo>*

Figure 10: Structure of the CMS SignedData component for the EF.SOD [ICA21b].

have adapted and included this process into our proof of concept.

In order to validate the data written to the chip, we have created a simple reader implementation also using the JMRTD library for processing the data to read the eMRTD. This application establishes secure communication between the reader and the chip and validates the content of the chip and prints the results into the console as output. As the first step, is crucial to validate the chain of trust, calculate and compare the hashes of the data groups and validate if the signature is issued by the given Document Signer. Both applications and their structure are referred to in the Appendix in Table 27 (EMRTD-Writer and EMRTD-Reader) and define our proof of concept for quantum resistant passive authentication process within the eMRTD.

We have created the eMRTD using our EMRTD-Writer application. For this eMRTD, we have used the combinations for the CSCA and Document Signer as shown in Table 20. This table also shows the size of the EF.SOD, which is measured in the Reader and the Writer applications. When the EF.SOD is signed using Falcon, the total size of the EF.SOD only takes up 4kB, whereas the Dilithium combination takes up two times more space on the eMRTD. We also included a SPHINCS+ combination to reflect on the findings in the previous sections. The result is that we were unable to measure the size of the EF.SOD on the chip for SPHINCS+ in the Reader application. In the Writer application we can see that the EF.SOD size is more than 47kB which is too large to be stored on the chip. These observations show that the lattice based candidates are most suitable to be used for passive authentication.

CSCA	**Document Signer**	**EF.SOD size**
Dilithium5 (V)	Dilithium2 (II)	9659 bytes
Falcon1024 (V)	Falcon512 (I)	4124 bytes
SPHINCS+ SHAKE-256s-simple (V)	SPHINCS+ SHAKE-192s-simple (III)	47384 bytes *

Table 20: Algorithms to be used for passive authentication for the eMRTD proof of concept. The SPHINCS+ entry is listed merely for comparison. Value of SPHINCS+ determined in our EMRTD-Writer application, as it could not be read on the chip.

8. Discussion

In our research we have investigated and benchmarked the three NIST standardization candidates, Dilithium, Falcon and SPHINCS+. A first observation is that the public and private key size of the lattice-based algorithms Dilithium and Falcon are larger than those of SPHINCS+. On the other hand however, we found that the signature size of the lattice-based signature algorithms Dilithium and Falcon are small when compared to the signature size of the SPHINCS+ variants. Another observation is that the key generation and signature generation process for SPHINCS+ is significantly longer than for Falcon and Dilithium. We find similar results for the benchmarks of the algorithms when used in X.509 certificates. These findings have been applied to our proof of concept and shows that an EF.SOD signed using a Falcon algorithm provides the smallest EF.SOD to be stored on the eMRTD, whereas the SPHINCS+ counterpart cannot be used in practice for passive authentication on a chip in the field.

Our research continues on the research of Pradel et al. in which they have used OpenSSL to generate and benchmark the PKI part for passive authentication [PM20]. Our research uses Bouncy Castle and JMRTD in order to generate the PKI for passive authentication and we have used the Java benchmarking framework JMH to benchmark the considered algorithms standalone and when used in certificates. In addition, we have established a post-quantum PKI structure for passive authentication which we have used in order to create and sign an eMRTD in practice. To validate if the eMRTD was written correctly, we have used JMRTD to create a simple reader application to validate the eMRTD and its contents by performing all required verification steps for passive authentication. Establishing the first quantum resistant eMRTD is a big next step in making travel documents future proof.

The research team is part of the Judicial Information Service, part of the Ministry of Justice and Security within the Kingdom of the Netherlands. This department is the expertise centre of chip technology and PKI for the Netherlands. Aside from being responsible for the CVCA and the authorisation of border terminals to privacy sensitive material on identity documents, the most important responsibility related to this research is the policy advisory role with regard to the CSCA of the Netherlands. We are additionally responsible for the NPKD functionality at the border in order to be able to perform passive authentication in practice. We also collaborate on European level and thus we are also represented in the Article 6 technical subgroup in the European Union in order to discuss and establish solutions for border related questions. On international level we are also represented in the ICAO PKD of the United Nations where we can collaborate on a worldwide level. We will share our findings on European level and on international level.

The quantum threat for passive authentication has a worldwide impact and establishing a baseline for passive authentication is a first required step in order to keep the borders safe in the future. This research contributes to this first step and opens new roads in order to start adopting and standardizing post-quantum algorithms for eMRTDs in the near future.

8.1. Limitations

In this research we have used JMH for benchmarking the algorithms within Java. Even though this framework is quite elaborate, it is still not a very known and thoroughly researched framework. In general it is quite difficult to cover all of the complexities of benchmarking in higher level programming languages. JMH does a good job in benchmarking the post-quantum algorithms as implemented by Bouncy Castle and gives a clear overview on the capabilities of the post-quantum algorithms in a practical situation. Still, more research on the functionality and reliability has to be performed on the JMH library. This is also important as there are not many other benchmarking frameworks for Java in the field and applications could benefit of benchmarking within Java in order to pinpoint problems or bottlenecks within higher level programming languages.

Our research is limited to the NIST standardization candidates, this narrows the choice of suitable cryptosystems for passive authentication. Currently we are waiting for the results of the call of proposals issued in 2022 by the NIST, this will take some time but can give useful insights on additional quantum resistant signature algorithms. The fact that only the lattice-based candidates are suitable for passive authentication is concerning, but could be overcome by another suitable candidate which makes use of other mathematical foundation. For SPHINCS+ we see differences in the simple and the robust implementations which show that the researchers are improving the algorithm to be more efficient. If the optimizations make it possible to reduce the size of the signature significantly, then SPHINCS+ could be considered again as candidate for passive authentication.

In this research we have used smartcards as they would be used in the field. We have taken into account that these smartcards are not publicly available on the web. Even though there are many security elements in place for the eMRTD, it would still make the eMRTD more accessible for passport forgers. This is also the reason why the proof of concept is specifically created for our research goals in order to keep a balance between making the research accessible to the community and keeping the passport forgers at bay. The security mechanisms themselves are described in this research, as they are meant to be publicly available in order to prevent security by obscurity. It could be possible to implement an eMRTD application from scratch, but this extra amount of work is far beyond the scope of this research.

The proof of concept for passive authentication favours the lattice-based signature algorithms Dilithium and Falcon. This is a risk to our results and the next steps for establishing a quantum resistant passive authentication protocol. The main risk here that there is always a possibility that a vulnerability or weakness is found in lattice-based signature algorithms which would make Dilithium and Falcon unusable. For passive authentication this would leave us empty handed and could jeopardize the safety of travel documents in the near future. As mentioned earlier, the call for proposals by the NIST could offer new algorithms. Our software has been taken into account during the implementation phase of this research. The proof of concept is publicly available and the software is designed in such a way to make the algorithms easily replaceable, as long as Bouncy Castle provides the implementation of the new algorithms. This makes it possible to continue our quest of providing a solution for passive authentication in a post-quantum era.

9. Conclusion

The evolving threat considering the upcoming of the quantum computer is not to be underestimated as it will impact IT infrastructures worldwide. Also for automatic border control, the quantum computer is a serious threat to be considered. Passive authentication is vulnerable for quantum attacks and the fact that passive authentication safeguards the integrity of travel documents, it is at risk in the near future. This research contributes to the rocky steep road for establishing a eMRTD which is resistant to quantum attacks using post-quantum algorithms. post-quantum signature algorithms however behave quite differently in contrast to the classical signature algorithms because they use different underlying mathematics. In our research we have evaluated and benchmarked the algorithms with in Java in order to determine which of these algorithms could be used in for passive authentication in the future.

Our benchmarks, considering the NIST standardization candidates within Java and the Bouncy Castle library, indicate that SPHINCS+ is not suitable to be used as replacement algorithm for passive authentication. The main reason is the large signature for the SPHINCS+ 256f/s and 192f variants of the algorithm. The eMRTD chip has limited storage capacity and these variants exceed or approach this limit leaving no space on the chip for other data. The lattice-based variants are more suitable when compared to SPHINCS+. The benchmark results show that the footprint of Falcon signatures is small and is followed closely by Dilithium. Also the key generation, signature generation and the signature validation processes of the lattice-based algorithms are executed very quickly when compared to the stateless hash based variants. Additionally, we have benchmarked the post-quantum algorithms when used during the creation and verification of X.509 certificates. The results of these benchmarks are consistent with those which only consider the algorithms themselves.

After benchmarking the post-quantum signature algorithms, we have established software for creating the post-quantum PKI for passive authentication and software for using this PKI in order to create a functional eMRTD. In order to validate the data, we have established additional software to validate the contents of the eMRTD by fully performing passive authentication. All software combined form our proof of concept and contribute to the developments of creating quantum resistant travel documents. We will share our findings with the European Article 6 Technical subgroup and the ICAO in order to keep our borders safe for now and the near future.

9.1. Future work

Passive authentication is not the only protocol which is at risk due to the upcoming threat of quantum computer. Extended Access Control (EAC), important for European passports, is vulnerable as well for the imminent quantum threat, since it also uses asymmetric cryptography. It is also mentioned in [PM20], but a practical implementation of EAC for a post-quantum era still has not been realised yet. It is interesting to elaborate further on this topic because EAC protects the privacy-sensitive data of the eMRTD from unauthorized reading [RS14], which is relevant for the passports issued within the European Union. In future research it is possible to explore the impact of quantum computing on EAC and how to mitigate its risks in order to protect the privacy-sensitive data on the passports. In addition to passive authentication and EAC, the remaining communication protocols such as PACE, AA and CA are vulnerable for quantum computers in the near future. In our research these protocols are considered a low risk when compared to passive authentication. Although it still remains a risk due to the presence of Key Encapsulation (KEM) and Key Exchange (KEX) mechanisms within these protocols, because these mechanisms are based on asymmetric cryptography. Future research could focus on improving the security of the communication protocols on chip environments and how to improve secure communication in a post-quantum era.

The public key validity period of a 10-year valid passport is 13 to 15 years [ICA21d]. In case of an overlap with the post-quantum era, this would mean that passports have to be retracted from citizens due to the usage of classical and non post-quantum resistant cryptography, which could be a costly operation. The transition process towards a post-quantum implementation of passive authentication is beyond the scope of this paper. Related research on hybrid certificates are interesting for this migration path, because these certificates act like ordinary X.509 certificates, but have an extension containing post-quantum algorithm related values [KPDG18]. Further investigation is required in order to determine if a migration path would benefit from using hybrid X.509 certificates in the migration to a quantum proof implementation of passive authentication. In addition to the technical challenges, there are also organizational challenges to overcome. Challenges such as standardizing the solution on a worldwide level in such a way that it aligns with the infrastructure and budget capabilities of the ICAO members. This requires a research on an organizational level to propose a solution that would be accepted and adopted by every member state of the ICAO.

In order to migrate to post-quantum cryptography, it is essential to have a clear overview of which algorithms are currently used in the field. For passive authentication for travel documents, it is currently unclear which classical algorithms are being used for the Document Signer certificates. A survey, in collaboration with governments or the ICAO could give more insights on which algorithms are currently used in the field and which certificates are already at risk without even considering the post-quantum threat. Including this research paper, it could be a good starting point of the discussion on how to migrate to post-quantum algorithms for travel documents and how to standardize these new algorithms.

The LDS for eMRTDs is also undergoing developments. LDS2 supports new applications such as digital Travel Records, Visa Records and Additional Biometrics applications [ICA21b]. These components also require separate PKI structures in order to guarantee the authenticity and integrity of the data stored in the new applications [ICA21c]. Our research only considers passive authentication for LDS1, however it is also interesting to expand the scope for LDS2 applications.

Finally, expanding the scope for the benchmarked algorithms is certainly worth investigating. It is very interesting to research stateful signature schemes such as XMSS and LMS as possible alternatives for passive authentication. The additional challenge here is to design such an infrastructure which is capable of handling stateful signature algorithms. It is interesting to research the best trade-off between the different parameter sets within XMSS and LMS using security and performance as distinctive variables. Using these algorithms for the PKI for passive authentication could provide useful new insights.

A. Bouncy Castle algorithm OID overview

Algorithm	Configuration	OID (BC)	Security Level
FALCON	Falcon-512	1.3.9999.3.1	I
Dilithium	Dilithium2	1.3.6.1.4.1.2.267.7.4.4	II
Dilithium	Dilithium2 (AES)	1.3.6.1.4.1.2.267.11.4.4	
Dilithium	Dilithium3	1.3.6.1.4.1.2.267.7.6.5	III
Dilithium	Dilithium3 (AES)	1.3.6.1.4.1.2.267.11.6.5	
Dilithium	Dilithium5	1.3.6.1.4.1.2.267.7.8.7	V
Dilithium	Dilithium5 (AES)	1.3.6.1.4.1.2.267.11.8.7	
FALCON	Falcon-1024	1.3.9999.3.4	

Table 21: Lattice based signature algorithms considered for benchmarking.

Algorithm	Configuration	OID (BC)	Security Level
SPHINCS+	SHA2-128s-robust	1.3.6.1.4.1.22554.2.5.1	I
SPHINCS+	SHA2-128f-robust	1.3.6.1.4.1.22554.2.5.2	
SPHINCS+	SHAKE-128s-robust	1.3.6.1.4.1.22554.2.5.3	
SPHINCS+	SHAKE-128f-robust	1.3.6.1.4.1.22554.2.5.4	
SPHINCS+	HARAKA-128s-robust	1.3.6.1.4.1.22554.2.5.5	
SPHINCS+	HARAKA-128f-robust	1.3.6.1.4.1.22554.2.5.6	
SPHINCS+	SHA2-128s-simple	1.3.6.1.4.1.22554.2.5.19	
SPHINCS+	SHA2-128f-simple	1.3.6.1.4.1.22554.2.5.20	
SPHINCS+	SHAKE-128s-simple	1.3.6.1.4.1.22554.2.5.21	
SPHINCS+	SHAKE-128f-simple	1.3.6.1.4.1.22554.2.5.22	
SPHINCS+	HARAKA-128s-simple	1.3.6.1.4.1.22554.2.5.23	
SPHINCS+	HARAKA-128f-simple	1.3.6.1.4.1.22554.2.5.24	
SPHINCS+	SHA2-192s-robust	1.3.6.1.4.1.22554.2.5.7	III
SPHINCS+	SHA2-192f-robust	1.3.6.1.4.1.22554.2.5.8	
SPHINCS+	SHAKE-192s-robust	1.3.6.1.4.1.22554.2.5.9	
SPHINCS+	SHAKE-192f-robust	1.3.6.1.4.1.22554.2.5.10	
SPHINCS+	HARAKA-192s-robust	1.3.6.1.4.1.22554.2.5.11	
SPHINCS+	HARAKA-192f-robust	1.3.6.1.4.1.22554.2.5.12	
SPHINCS+	SHA2-192s-simple	1.3.6.1.4.1.22554.2.5.25	
SPHINCS+	SHA2-192f-simple	1.3.6.1.4.1.22554.2.5.26	
SPHINCS+	SHAKE-192s-simple	1.3.6.1.4.1.22554.2.5.27	
SPHINCS+	SHAKE-192f-simple	1.3.6.1.4.1.22554.2.5.28	
SPHINCS+	HARAKA-192s-simple	1.3.6.1.4.1.22554.2.5.29	
SPHINCS+	HARAKA-192f-simple	1.3.6.1.4.1.22554.2.5.30	
SPHINCS+	SHA2-256s-robust	1.3.6.1.4.1.22554.2.5.13	V
SPHINCS+	SHA2-256f-robust	1.3.6.1.4.1.22554.2.5.14	
SPHINCS+	SHAKE-256s-robust	1.3.6.1.4.1.22554.2.5.15	
SPHINCS+	SHAKE-256f-robust	1.3.6.1.4.1.22554.2.5.16	
SPHINCS+	HARAKA-256s-robust	1.3.6.1.4.1.22554.2.5.17	
SPHINCS+	HARAKA-256f-robust	1.3.6.1.4.1.22554.2.5.18	
SPHINCS+	SHA2-256s-simple	1.3.6.1.4.1.22554.2.5.31	
SPHINCS+	SHA2-256f-simple	1.3.6.1.4.1.22554.2.5.32	
SPHINCS+	SHAKE-256s-simple	1.3.6.1.4.1.22554.2.5.33	
SPHINCS+	SHAKE-256f-simple	1.3.6.1.4.1.22554.2.5.34	
SPHINCS+	HARAKA-256s-simple	1.3.6.1.4.1.22554.2.5.35	
SPHINCS+	HARAKA-256f-simple	1.3.6.1.4.1.22554.2.5.36	

Table 22: SPHINCS+ configurations considered for benchmarking. There are 36 different configurations due to the characteristics of SPHINCS+.

B. Best suited algorithm extended results

Algorithm	Private Key	Public Key	Signature	Sec. Level
Falcon-512	2223	915	* 655	I
SPHINCS+ HARAKA-128f-robust	101	58	17088	
SPHINCS+ HARAKA-128s-robust	101	58	7856	
SPHINCS+ HARAKA-128f-simple	101	58	17088	
SPHINCS+ HARAKA-128s-simple	101	58	7856	
SPHINCS+ SHA2-128f-robust	101	58	17088	
SPHINCS+ SHA2-128s-robust	101	58	7856	
SPHINCS+ SHA2-128f-simple	101	58	17088	
SPHINCS+ SHA2-128s-simple	101	58	7856	
SPHINCS+ SHAKE-128f-robust	101	58	17088	
SPHINCS+ SHAKE-128s-robust	101	58	7856	
SPHINCS+ SHAKE-128f-simple	101	58	17088	
SPHINCS+ SHAKE-128s-simple	101	58	7856	
Dilithium2	3902	1336	2420	II
Dilithium2 (AES)	3902	1336	2420	
Dilithium3	6014	1976	3293	III
Dilithium3 (AES)	6014	1976	3293	
SPHINCS+ HARAKA-192f-robust	134	74	35664	
SPHINCS+ HARAKA-192s-robust	134	74	16224	
SPHINCS+ HARAKA-192f-simple	134	74	35664	
SPHINCS+ HARAKA-192s-simple	134	74	16224	
SPHINCS+ SHA2-192f-robust	134	74	35664	
SPHINCS+ SHA2-192s-robust	134	74	16224	
SPHINCS+ SHA2-192f-simple	134	74	35664	
SPHINCS+ SHA2-192s-simple	134	74	16224	
SPHINCS+ SHAKE-192f-robust	134	74	35664	
SPHINCS+ SHAKE-192s-robust	134	74	16224	
SPHINCS+ SHAKE-192f-simple	134	74	35664	
SPHINCS+ SHAKE-192s-simple	134	74	16224	
Dilithium5	7518	2616	4595	V
Dilithium5 (AES)	7518	2616	4595	
Falcon-1024	4143	1811	* 1271	
SPHINCS+ HARAKA-256f-robust	168	90	49856	
SPHINCS+ HARAKA-256s-robust	168	90	29792	
SPHINCS+ HARAKA-256f-simple	168	90	49856	
SPHINCS+ HARAKA-256s-simple	168	90	29792	
SPHINCS+ SHA2-256f-robust	168	90	49856	
SPHINCS+ SHA2-256s-robust	168	90	29792	
SPHINCS+ SHA2-256f-simple	168	90	49856	
SPHINCS+ SHA2-256s-simple	168	90	29792	
SPHINCS+ SHAKE-256f-robust	168	90	49856	
SPHINCS+ SHAKE-256s-robust	168	90	29792	
SPHINCS+ SHAKE-256f-simple	168	90	49856	
SPHINCS+ SHAKE-256s-simple	168	90	29792	

Table 23: Storage-based properties of PQC algorithms grouped per security level. Results of the measurements are in bytes. (*) Falcon has a varying signature length. The listed result is the average size over 100 iterations.

Algorithm	**Key generation**	**Sign. generation**	**Sign. validation**	**Sec. Level**
Falcon-512	15396	1323	74	
SPHINCS+ HARAKA-128f-robust	10289	256425	16302	
SPHINCS+ HARAKA-128s-robust	657620	5849434	6215	
SPHINCS+ HARAKA-128f-simple	6258	156738	9675	
SPHINCS+ HARAKA-128s-simple	399980	3590335	3701	
SPHINCS+ SHA2-128f-robust	5815	140844	8545	
SPHINCS+ SHA2-128s-robust	373859	3150242	2911	I
SPHINCS+ SHA2-128f-simple	2789	67842	3900	
SPHINCS+ SHA2-128s-simple	174893	1525860	1357	
SPHINCS+ SHAKE-128f-robust	7350	177297	10506	
SPHINCS+ SHAKE-128s-robust	465832	3987983	3570	
SPHINCS+ SHAKE-128f-simple	3822	93563	5297	
SPHINCS+ SHAKE-128s-simple	243217	2063663	1803	
Dilithium2	158	563	170	II
Dilithium2 (AES)	290	737	279	
Dilithium3	277	1011	264	
Dilithium3 (AES)	529	1278	538	
SPHINCS+ HARAKA-192f-robust	15223	451000	24583	
SPHINCS+ HARAKA-192s-robust	972938	11086931	9513	
SPHINCS+ HARAKA-192f-simple	595284	6627328	5357	
SPHINCS+ HARAKA-192s-simple	595284	6627328	5357	
SPHINCS+ SHA2-192f-robust	8442	225112	12277	
SPHINCS+ SHA2-192s-robust	548427	5406656	4372	III
SPHINCS+ SHA2-192f-simple	4016	108477	5653	
SPHINCS+ SHA2-192s-simple	260883	2706109	2023	
SPHINCS+ SHAKE-192f-robust	10894	284485	15482	
SPHINCS+ SHAKE-192s-robust	684438	6730109	5345	
SPHINCS+ SHAKE-192f-simple	5586	148839	7747	
SPHINCS+ SHAKE-192s-simple	357170	3568502	2613	
Dilithium5	428	1196	436	
Dilithium5 (AES)	859	1693	862	
Falcon-1024	42182	2745	148	
SPHINCS+ HARAKA-256f-robust	40785	968036	26352	
SPHINCS+ HARAKA-256s-robust	646524	10647582	14274	
SPHINCS+ HARAKA-256f-simple	24649	585561	15241	
SPHINCS+ HARAKA-256s-simple	390676	6442313	8249	
SPHINCS+ SHA2-256f-robust	29642	605148	16539	V
SPHINCS+ SHA2-256s-robust	479833	5885411	8296	
SPHINCS+ SHA2-256f-simple	10605	227719	5796	
SPHINCS+ SHA2-256s-simple	173780	2357170	2997	
SPHINCS+ SHAKE-256f-robust	28749	589743	15820	
SPHINCS+ SHAKE-256s-robust	457504	5657210	7815	
SPHINCS+ SHAKE-256f-simple	14692	314625	7877	
SPHINCS+ SHAKE-256s-simple	238113	3055240	3877	

Table 24: Performance-based properties of PQC algorithms using the JMH framework in Java and are grouped per security level. Results are the average time needed to execute the corresponding Bouncy Castle function (in μs).

C. X.509 PQC BENCHMARK RESULTS

Algorithm	Certificate Size	Security Level
Falcon-512	2182	I
HARAKA-128f-robust	17768	
HARAKA-128s-robust	8536	
HARAKA-128f-simple	17768	
HARAKA-128s-simple	8536	
SHA2-128f-robust	17768	
SHA2-128s-robust	8536	
SHA2-128f-simple	17768	
SHA2-128s-simple	8536	
SHAKE-128f-robust	17768	
SHAKE-128s-robust	8536	
SHAKE-128f-simple	17768	
SHAKE-128s-simple	8536	
Dilithium2	4380	II
Dilithium2 (AES)	4380	
Dilithium3	5893	III
Dilithium3 (AES)	5893	
HARAKA-192f-robust	36360	
HARAKA-192s-robust	16920	
HARAKA-192f-simple	36360	
HARAKA-192s-simple	16920	
SHA2-192f-robust	36360	
SHA2-192s-robust	16920	
SHA2-192f-simple	36360	
SHA2-192s-simple	16920	
SHAKE-192f-robust	36360	
SHAKE-192s-robust	16920	
SHAKE-192f-simple	36360	
SHAKE-192s-simple	16920	
Dilithium5	7835	V
Dilithium5 (AES)	7835	
Falcon-1024	3694	
HARAKA-256f-robust	50568	
HARAKA-256s-robust	30504	
HARAKA-256f-simple	50568	
HARAKA-256s-simple	30504	
SHA2-256f-robust	50568	
SHA2-256s-robust	30504	
SHA2-256f-simple	50568	
SHA2-256s-simple	30504	
SHAKE-256f-robust	50568	
SHAKE-256s-robust	30504	
SHAKE-256f-simple	50568	
SHAKE-256s-simple	30504	

Table 25: Results of the benchmark on the storage properties of the X.509 certificates using the Bouncy Castle library (in bytes).

Algorithm	**Generate CSR**	**Create CSCA**	**Validation**	**Sec. Level**
Falcon-512	1353	1341	102	I
HARAKA-128f-robust	256644	257675	16359	
HARAKA-128s-robust	5845423	5853879	6262	
HARAKA-128f-simple	158303	157936	9707	
HARAKA-128s-simple	3589672	3575960	3734	
SHA2-128f-robust	140132	140696	8511	
SHA2-128s-robust	3149854	3155434	2923	
SHA2-128f-simple	68028	67796	3932	
SHA2-128s-simple	1531751	1523185	1381	
SHAKE-128f-robust	176194	175848	10520	
SHAKE-128s-robust	4004197	4000166	3600	
SHAKE-128f-simple	92770	92481	5278	
SHAKE-128s-simple	2087836	2097904	1817	
Dilithium2	657	631	187	II
Dilithium2 (AES)	817	729	303	
Dilithium3	1048	1024	291	III
Dilithium3 (AES)	1324	1342	508	
HARAKA-192f-robust	450579	449633	24592	
HARAKA-192s-robust	11035614	11122161	9479	
HARAKA-192f-simple	268422	268640	14386	
HARAKA-192s-simple	6667497	6633352	5450	
SHA2-192f-robust	232570	232376	12762	
SHA2-192s-robust	5557838	5569566	4519	
SHA2-192f-simple	109114	108183	5836	
SHA2-192s-simple	2706095	2691460	2055	
SHAKE-192f-robust	282593	283175	15465	
SHAKE-192s-robust	6739412	6732299	5348	
SHAKE-192f-simple	149992	148925	7702	
SHAKE-192s-simple	3579142	3584922	2663	
Dilithium5	1145	1291	466	V
Dilithium5 (AES)	1659	1712	897	
Falcon-1024	2773	2764	184	
HARAKA-256f-robust	965837	961192	26308	
HARAKA-256s-robust	10588482	10596983	14301	
HARAKA-256f-simple	587901	589847	15442	
HARAKA-256s-simple	6480646	6493376	8345	
SHA2-256f-robust	628469	618112	16944	
SHA2-256s-robust	5917595	5903589	8445	
SHA2-256f-simple	236426	235911	6044	
SHA2-256s-simple	2351470	2358679	3027	
SHAKE-256f-robust	590905	592943	15736	
SHAKE-256s-robust	5701171	5705696	7813	
SHAKE-256f-simple	315383	315454	7990	
SHAKE-256s-simple	3060669	3090921	3934	

Table 26: Results of the benchmark for the performance properties of the X.509 certificates using the Bouncy Castle library (in μs).

D. Related software

Project	**Description**	**URL**
PQC-BC-Benchmark	JMH Benchmark application which can be used to benchmark the time related properties of the considered PQC algorithms.	https://gitlab.com/pqc-research/pqc-bc-benchmark
PQC-BC-Benchmark-Keysizes	Application which is used to benchmark the storage related properties of the considered PQC algorithms.	https://gitlab.com/pqc-research/pqc-bc-keysizes
PQC-BC-Wrapper	Dependency of PQC-BC-Benchmark (both applications). Wrapper library which makes it easier to instantiate the PQC algorithms in Bouncy Castle.	https://gitlab.com/pqc-research/pqc-bc-lib
Masterlist Tool	Simple command line tool to analyse the algorithms used for the CSCA in three different masterlists	https://gitlab.com/pqc-research/masterlist-tool
EMRTD-Writer	Application which is responsible for creating the PQ PKI and uses this PKI to create an quantum resistant eMRTD.	https://gitlab.com/pqc-research/emrtd-writer
EMRTD-Reader	Application using the JMRTD library (as much as possible) to verify if the contents which are stored on the chip are valid.	https://gitlab.com/pqc-research/emrtd-reader
APDU-LIB	Dependency of the EMRTD-Reader and EMRTD-writer for shared functionality.	https://gitlab.com/pqc-research/apdu-lib

Table 27: The source code of the software which has been established and used during this research. All software was compiled using Adoptium JDK 17 and Maven. IDE used was IntelliJ IDEA.

References

[AAC$^+$22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the third round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2022. 9, 10, 29, 30

[AASA$^+$19] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the first round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2019. 9

[AASA$^+$20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020. 9, 10

[ABB$^+$22] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. Sphincs+. *Submission to the NIST post-quantum project, v.3.1*, Jun 2022. 30, 34

[Bar20] Elaine Barker. Recommendation for: Key management part 1 - general. Technical Report SP 800-57, U.S. Department of Commerce, Washington, D.C., May 2020. 43

[BDH11] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - a practical forward secure signature scheme based on minimal security assumptions. In *Post-Quantum Cryptography*, volume 2011, pages 117–129, Nov 2011. 9, 10, 31

[BDK$^+$21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium algorithm specifications and supporting documentation (version 3.1). *Submission to the NIST's post-quantum cryptography standardization process*, Feb 2021. 30, 34, 35

[Ber09] Daniel Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete. In *SHARCS'09 Workshop Record (Proceedings*

4th Workshop on Special-purpose Hardware for Attacking Cryptograhic Systems, Lausanne, Switserland, September 9-10, 2009), pages 105–116, Jan 2009. 8

[Beu22] Ward Beullens. Breaking Rainbow takes a weekend on a laptop. Cryptology ePrint Archive, Paper 2022/214, 2022. https://eprint.iacr.org/2022/214. 9

[BHH+15] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In *Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34*, pages 368–397. Springer, 2015. 30, 35

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2129–2146, New York, NY, USA, 2019. Association for Computing Machinery. 30

[BHL22] Daniel J. Bernstein, Andreas Hülsing, and Tanja Lange. Post-quantum cryptography - integration study, Oct 2022. 12, 30, 32

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN'98: Theoretical Informatics*, pages 163–169. Springer Berlin Heidelberg, 1998. 8

[BKM+14] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and John M. Martinis. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, Apr 2014. 10

[BR19] Elaine Barker and Allen Roginsky. Transitioning the use of cryptographic algorithms and key lengths. Technical Report 800-131A Rev. 2, National Institute of Standards and Technology, Washington, D.C., Mar 2019. 13, 14

[BSI15] BSI. Advanced security mechanisms for machine readable travel documents and eIDAS token - Part 1 – eMRTDs with BAC/PACEv2 and EACv1.

Technical Guideline TR-03110-1, Bundesamt für Sicherheit in der Informationstechnik (BSI), Postfach 20 03 63 53133 Bonn, 2015. 22, 24

[BSKNS20] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. Post-quantum TLS on embedded systems. Cryptology ePrint Archive, Paper 2020/308, 2020. https://eprint.iacr.org/2020/308. 11

[BSP+08] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008. 21

[CAD+20] David A. Cooper, Daniel C. Apon, Quynh H. Dang, Michael S. Davidson, Morris J. Dworkin, and Carl A. Miller. Recommendation for stateful hash-based signature schemes. Technical Report SP 800-208, U.S. Department of Commerce, Washington, D.C., Oct 2020. 9, 31

[CBL+21] Diego Costa, Cor-Paul Bezemer, Philipp Leitner, Artur Andrzejak, IT-fakulteten, Göteborgs universitet, Gothenburg University, IT Faculty, and Software Engineering (GU) Institutionen för data-och informationsteknik. What's wrong with my benchmark results? Studying bad practices in JMH benchmarks. *IEEE transactions on software engineering*, 47(7):1452–1467, 2021. 36

[CD22] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975. 9

[CML22] Lily Chen, Dustin Moody, and Yi-Kai Liu. Call for proposals - post-quantum cryptography: Digital signature schemes: CSRC, Aug 2022. https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals, Accessed on 2023-02-19. 29

[CN22] Hugh Collins and Chris Nay. IBM unveils 400 qubit-plus quantum processor and next-generation IBM quantum system two, Nov 2022. https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two, Accessed on 2023-02-19. 10

[DD92] George I. Davida and Yvo G. Desmedt. Passports and visas versus IDs. *Computers & Security*, 11(3):253–258, 1992. 6

[FHK+20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process*, oct 2020. 30, 35, 39

[FMMC12] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), Sep 2012. 10

[GE21] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, Apr 2021. 10

[Glo18] GlobalPlatform. Card specification version 2.3.1, March 2018. Available: `https://globalplatform.org/wp-content/uploads/2018/05/GPC_CardSpecification_v2.3.1_PublicRelease_CC.pdf`. 46

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. 7, 14, 17, 24

[GW22] Ruben Gonzalez and Thom Wiggers. KEMTLS vs. post-quantum TLS: Performance on embedded systems. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 99–117, Cham, 2022. Springer Nature Switzerland. 11

[HBG+18] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, May 2018. 31

[Hou02] Russ Housley. Cryptographic Message Syntax (CMS). RFC 3369, Sep 2002. 45

[ICA21a] ICAO. Doc 9303: Machine Readable Travel Documents - Part 1: Introduction. Technical report 8th Edition, International Civil Aviation Organisation (ICAO), 999 Robert-Bourassa Boulevard, Montréal, Quebec, Canada H3C 5H7, Nov 2021. 6

[ICA21b] ICAO. Doc 9303: Machine Readable Travel Documents - Part 10: Logical Data Structure (LDS) for Storage of Biometrics and Other Data in the Contactless Integrated Circuit (IC). Technical report 8th Edition, International Civil Aviation Organisation (ICAO), 999 Robert-Bourassa Boulevard, Montréal, Quebec, Canada H3C 5H7, 2021. 6, 19, 20, 45, 47, 54

[ICA21c] ICAO. Doc 9303: Machine Readable Travel Documents - Part 11: Security Mechanisms for MRTDs. Technical report 8th Edition, International Civil Aviation Organisation (ICAO), 999 Robert-Bourassa Boulevard, Montréal, Quebec, Canada H3C 5H7, 2021. 6, 7, 8, 11, 13, 14, 15, 16, 17, 18, 21, 22, 23, 24, 25, 26, 27, 54

[ICA21d] ICAO. Doc 9303: Machine Readable Travel Documents - Part 12: Public Key Infrastructure for MRTDs. Technical report 8th Edition, International Civil Aviation Organisation (ICAO), 999 Robert-Bourassa Boulevard, Montréal, Quebec, Canada H3C 5H7, 2021. 6, 19, 20, 40, 53

[ISO20] ISO. Identification cards — integrated circuit cards — part 4: Organization, security and commands for interchange. Standard, International Organization for Standardization, Geneva, Switzerland, May 2020. 45

[KF17] Panos Kampanakis and Scott Fluhrer. Lms vs xmss: Comparion of two hash-based signature standards. *Cryptology ePrint Archive*, 2017. 31, 32

[KPDG18] Panos Kampanakis, Peter Panburana, Ellie Daw, and Daniel Van Geest. The viability of post-quantum X.509 certificates. Cryptology ePrint Archive, Paper 2018/063, 2018. `https://eprint.iacr.org/2018/063`. 53

[LM95] Frank T Leighton and Silvio Micali. Large provably fast and secure digital signature schemes based on secure hash functions, Jul 1995. US Patent 5,432,852. 9, 31

[MCF19] David McGrew, Michael Curcio, and Scott Fluhrer. Leighton-Micali Hash-Based Signatures. RFC 8554, Apr 2019. 31

[McH21] Jenny M McHugh. The passport's medieval forebear: Grants of safe-conduct in medieval Britain, Nov 2021. Accessed on 2023-02-12. Available: `https://www.epoch-magazine.com/post/the-passport-s-medieval-forebear-grants-of-safe-conduct-in-medieval-britain`. 6

[MCJ$^+$16] Dustin Moody, Lily Chen, Stephen Jordan, Yi-Kai Liu, Daniel Smith, Ray Perlner, and René Peralta. NIST report on post-quantum cryptography. Technical report, U.S. Department of Commerce, Apr 2016. 8, 28

[Mer79] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1979. AAI8001972. 31

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York. 31

[MK19] Soundes Marzougui and Juliane Krämer. Post-quantum cryptography in embedded systems. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA, 2019. Association for Computing Machinery. 11

[Moo18] Dustin Moody. Let's get ready to rumble. the NIST PQC "competition". In *Proc. of First PQC Standardization Conference*, pages 11–13, 2018. 35

[Mos18] M. Mosca. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy*, 16(05):38–41, Sep 2018. 8, 28

[MPD$^+$18] Lukas Malina, Lucie Popelova, Petr Dzurenda, Jan Hajny, and Zdenek Martinasek. On feasibility of post-quantum cryptography on small devices. *IFAC-PapersOnLine*, 51(6):462–467, 2018. 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018. 11

[MS22] Soundes Marzougui and Jean-Pierre Seifert. XMSS-based chain of trust. In Ulrich Kühne and Fan Zhang, editors, *Proceedings of 10th International Workshop on Security Proofs for Embedded Systems*, volume 87 of *EPiC Series in Computing*, pages 66–82. EasyChair, 2022. 10

[MVZJ18] Vasileios Mavroeidis, Kamer Vishi, Mateusz D. Zych, and Audun Jøsang. The impact of quantum computing on present cryptography. *CoRR*, abs/1804.00200, 2018. 8, 18, 20, 22, 26

[oST17] National Institute of Standards and Technology. Recommendation for the triple data encryption algorithm (TDEA) block cipher. Technical Report SP 800-67 Rev. 2, U.S. Department of Commerce, Washington, D.C., 2017. 13, 18

[PM20] Gaëtan Pradel and Chris J. Mitchell. *Post-quantum Certificates for Electronic Travel Documents*, pages 56–73. Computer Security. Springer International Publishing, Cham, 2020. 9, 10, 49, 53

[RMYK17] Sandeep Rao, Dindayal Mahto, DILIP YADAV, and Danish Khan. The AES-256 cryptosystem resists quantum attacks. *International Journal of Advanced Research in Computer Science*, 8:404–408, Apr 2017. 8, 17

[RS14] Antonia Rana and Luigi Sportiello. Implementation of security and privacy in ePassports and the extended access control infrastructure. *International Journal of Critical Infrastructure Protection*, 7(4):233–243, 2014. 7, 53

[Shi13] Aleksey Shipilev. The art of javabenchmarking. *Aleksey Shipilёv: one-stop page*, 2013. 36

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997. 6, 7, 18, 20, 22, 24, 26

[TDF$^+$23] George Tasopoulos, Charis Dimopoulos, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. Energy consumption evaluation of post-quantum tls 1.3 for resource-constrained embedded devices. Cryptology ePrint Archive, Paper 2023/506, 2023. `https://eprint.iacr.org/2023/506`. 11

[TLF$^+$22] George Tasopoulos, Jinhui Li, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. *Performance Evaluation of Post-Quantum TLS 1.3 on Resource-Constrained Embedded Systems*, pages 432–451. Information Security Practice and Experience. Springer International Publishing, Cham, 2022. 11