A Survey of Digital Signing in the Post Quantum Era

Teik Guan Tan

ISTD, Singapore University of Technology and Design, Singapore, teikguan_tan@mymail.sutd.edu.sg

Jianying Zhou

ISTD, Singapore University of Technology and Design, Singapore, jianying_zhou@sutd.edu.sg

ABSTRACT

Public key cryptography is threatened by the advent of quantum computers. Using Shor's algorithm on a large-enough quantum computer, an attacker could cryptanalyze any RSA/ECDSA public key, and generate fake digital signatures in seconds. In this paper, we profile all 9 digital signature candidate algorithms within NIST's post-quantum cryptography contest round 2, plus stateful hash-based signatures, and evaluate their suitability against 11 different industry applications. We have found that Falcon, a lattice-based digital signing algorithm, when supplemented with XMSS/LMS hash-based signatures, can best meet all of the application requirements if improvements in key generation and key sizes are achieved.

CCS CONCEPTS

• Security and privacy→Cryptography→Public Key (asymmetric) techniques→Digital Signatures • Security and privacy→Cryptography→Cryptanalysis and other attacks

KEYWORDS

Digital Signing, Post Quantum Cryptography, Public Key Cryptography

1 Introduction

The use of asymmetric key cryptography to create digital signatures as a means of data/entity authentication or transaction non-repudiation is pervasive. On any given day, millions of web servers use certificates and digital signing as part of the Transport Layer Security (TLS) [1, 2] to allow users to verify the identity of the server. Millions of merchant payment terminals verify the digital signatures from EMV [3] payment cards to ascertain that the cards were not cloned. And phone manufacturers rely on digital signatures to protect the integrity of the operating system software and applications running on the millions of mobile phones.

The use of digital signatures is also critical for the legal validity of electronic documents. In the European Union, the eIDAS (Electronic Identification, Authentication and trust Services) regulation N° 910/2014 [4] recognizes the use of digital signing for qualified electronic signatures (QES) to attain a "high" level of assurance and equates it to the legal equivalent of a hand-written signature. Similarly, the ESIGN (Electronic Signatures in Global and National Commerce) Act 2000 [5] in United States allows for the recognition of electronic documents as legal provided, amongst other requirements, that the electronic document is "retained and accurately reproduced for later reference", a property that can be achieved using digital signatures.

Most digital signatures are implemented using RSA [6] or elliptic curve cryptography (ECC) [7]. To sign a message, the signer (or prover) has in sole possession a secret (or private) key as well as the corresponding public key which is published. The signer uses the secret key to encrypt the message, or more accurately the hash of the message, to generate the signature, and any recipient (or verifier) can use the signer's public key and signature to verify that the hash of the message was indeed encrypted by signer's secret key (non-repudiation property). If the message was modified in any way, the signature verification would fail (integrity property).

Yet, RSA and ECC based digital signature implementations face a potential catastrophic vulnerability in the face of quantum computers. Quantum computers work differently from the classical computers which operate on the principle of distinct bits 0 and 1. Quantum computers use the concept of a qubit (quantum bit) where each qubit can exist in a state of superposition, hovering between states 0 and 1 during computation, until the qubit is finally accessed. During computation, quantum algorithms (https://quantumalgorithmzoo.org/) use quantum gates or circuits to make qubits interact with each other using interference and entanglement to fundamentally achieve computational speed-ups that is not possible in classical computers. Both RSA and ECC have been proven theoretically to be vulnerable to quantum computers. Using Shor's algorithm [8, 9], an attacker in possession of a large quantum computer (of over 4000 qubits) can cryptanalyze the RSA or ECC public key and obtain the corresponding RSA or ECC secret key in O(logN) time. On one hand, quantum computer development is still at its infancy and the largest quantum computer commercially available has only 20 qubits [10] with a 53-qubit quantum computer soon to be available [11]. But this is set to change with the large amounts of money put into quantum computing research. National Institute of Science and Technology (NIST) stated in its 2016 report [12] that with a budget of 1 billion dollars, an RSA-2048 bit key can likely be broken by a large quantum computer in a matter of hours.

Another quantum algorithm of note is Grover's algorithm [13]. While Grover's algorithm was not created to specifically perform cryptanalysis, it is able to provide a quadratic speed-up in performance over an unsorted search space. This means that using Grover's algorithm in a quantum computer, an attacker can reduce the computational complexity in a brute-force search to $O(\sqrt{N})$. With respect to digital signatures, Grover's algorithm is typically applied to find the preimage of hashes where an attacker may attempt to find another message that corresponds to the same hash that was encrypted by the signer's secret key. If the hash has a security bit strength of 256-bits, Grover's algorithm on a large quantum computer will reduce the hashing bit-strength to 128. The good news is that this makes hashing still relatively-resistant to quantum computer attacks since we can linearly increase the bit-strength of hashing to defend against Grover's algorithm. The better news is that $O(\sqrt{N})$ complexity achieved by Grover's algorithm for brute-force search is proven to be optimal even on quantum computers [14]. This effectively puts hash-based cryptography (Section 2.1.4) as the only algorithm that is provably resistant to quantum computers.

The objective of this paper is to study the readiness and completeness of the work by the researchers and industry in preparation for the post-quantum (PQ) era where existing and new digital signature implementations remain secure despite the existence of quantum computers. In Section 2, we investigate the relevant digital signature algorithms, paying particular attention to the NIST Post Quantum standardization efforts [15]. In Section 3, we introduce our model to profile the readiness and completeness of the existing digital signature algorithms based on various signature characteristics. In Section 4, we apply the model to various industry systems that rely on digital signing as part of the application use-case and score the technological or implementational gaps. In Section 5, we analyze the gaps to identify the more promising algorithms and combination of algorithms. Finally, in Section 6, we conclude our findings and list the actions yet taken to prepare for the post-quantum era.

2 Background

NIST (National Institute of Science and Technology) has already started to solicit a Post-Quantum Cryptography (PQC) candidate in preparation for post-quantum readiness. The goal would be to update FIPS 186-4 [16], which defines the digital signature standard to be used to protect the integrity and non-repudiation of applications and data, in the next 3-5 years [12]. In Moody's presentation [15] in 2017, he gave an update on the ongoing NIST Post-Quantum Cryptography contest where he mentioned that PQC is more complicated and this effort was more of a standardization plan rather than a contest as they did not expect to "pick a winner". Good algorithms exist, but each has some disadvantages. For example, hash-based cryptography is quantum-resistant [13, 14], but the limited number of signatures per private key using hash-based signatures [17] and the relative large overheads in signing may make it impractical in some application use cases such as for signing emails and secure browsing. Is it therefore possible to select a suite of signature schemes that can fulfil all the digital signing requirements presently used by applications in the field? Or are we faced with a technology gap where existing applications will be left stranded without a viable digital signing solution in the face of quantum computers?

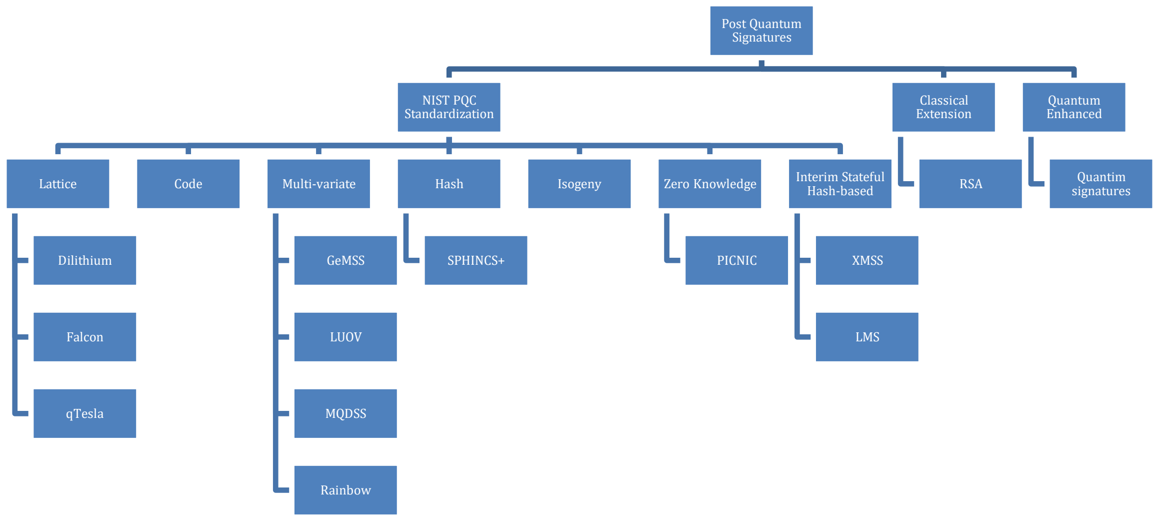

Figure 1. Landscape of post-quantum digital signing algorithms

As illustrated in Figure 1, this paper will first focus on the standardization efforts by NIST since it continues to garner significant efforts from the community at large to work on the problem, while providing results in a transparent and neutral way. Beyond NIST's PQC standardization, we also explore extensions to classical algorithms as well as quantum-enhanced algorithms (i.e. using quantum technology to supplement classical systems) [18] in their ability to provide post-quantum digital signatures.

2.1 NIST PQC Standardization

We start by looking at the evaluation criteria in NIST's PQC contest [19, 20]. NIST has taken a top-down functional approach to identifying the appropriate requirements for standardizing the new suite of PQC algorithms. They are broadly categorized into Security, Cost and Simplicity in terms of priority:

1. Security - NIST requires that the new PQC algorithms remain secure against both classical and quantum adversaries. NIST has defined 5 levels of security categories listed in Table 1 where submissions had to be benchmarked against.

Table 1. 5 levels of security categories for algorithms to benchmark against [14]

Level	Security Description
I	At least as hard to break as AES128 (exhaustive key search)
II	At least as hard to break as SHA256 (collision search)
III	At least as hard to break as AES192 (exhaustive key search)
IV	As least as hard to break as SHA384 (collision search)
V	At least as hard to break as AES256 (exhaustive key search)

Other security criteria include resistance to side-channel attacks [21] and achieving perfect forward secrecy (PFS) [22].

2. Costs - Costs in this case does not refer to a direct monetary value, but the amount of resources needed to operate the signature scheme. These could be in the area of key and signature sizes, computational overheads to perform key generation and signing. Intuitively, less costs would mean greater acceptance by more applications, but that may translate to a lower level of security. Is this compromise equitable? Will there exist a situation where costs are too prohibitive for applications even to maintain at the minimum level 1 (128-bit security strength) category?

3. Algorithm/Implementation characteristics – This refers to how easy it is for the new PQC algorithm to be implemented for a new system or as a replacement for an existing deployment. Considerations

include the flexibility of the algorithm to support digital signing versus public key encryption or key exchange, the resistance to algorithm misuse and the possibility of a drop-in replacement for popular security libraries such as Transport Layer Security (TLS) [2, 23].

Of the 82 submissions that were received by NIST, 69 were shortlisted for round 1 and a further shortlist narrowed the field to 26 second round candidates [20]. Of the 26, there are only a total of 9 candidate signature schemes (the remaining 17 are key exchange algorithms) which we briefly explore below. We also include stateful hash-based digital signatures which is an interim standard worked on by NIST with IETF.

2.1.1 Lattice-based Cryptography

Lattice-based cryptography has by far gained the most attention as the potential PQC candidate to address the threat of quantum computers. We believe that the interest is due to the flexibility of the lattice structure to support both encryption and digital signing well, as well as the reasonable key and signature sizes of 1-3kbytes for a 128-bit security strength. Of all the categories classified under the NIST PQC standardization, only lattice-based cryptography has 2nd round candidates in both key exchange and digital signing.

Lattice-based cryptography is a class of cryptographic primitives built on a multi-dimensional lattice structure and first used by Ajtai in 1996 [24]. Ajtai presented a cryptographic construction using lattices based on the short integer solution (SIS) problem and showed it secure in the average case if the shortest vector problem (SVP) was hard in the worst case. Goldreich, Goldwasser and Halevi (GGH) [25] introduced a more practical variant based on the lattice-reduction or closest vector problem (CVP) but was broken by [26]. GGH eventually gave rise to other SVP lattice algorithms such as NTRU (N^{th} degree Truncated Polynomial Ring Unit) [27]. Regev [28] used a different hard problem based on learning with errors (LWE) problem and showed that it was similarly secure if lattice problems were hard in the worst case. Ring-learning with errors (RLWE) has also emerged as a strong variant of LWE with practical smaller key size properties [29]. Using SVP and its variants, Gentry, Peikert and Vaikuntanathan (GPV) [30] formalized a provably secure (over classical and quantum oracle models) hash-and-sign approach to digital signing over lattices, broadly described below:

1. The message to be signed is first hashed to a position on the lattice.
2. The signer can then use the trapdoor, or short basis (i.e. the secret key), to find a short vector on the lattice that is close to the hash and does not statistically leak the private key, and publish this as the signature.
3. The verifier first checks that the signature vector is short, and then performs the same hash and uses the public key (which is the long basis of the lattice) to verify that the signature is close to the hash.

Different variants of trapdoor-based signing optimize for performance and security by using specialized lattices (e.g. NTRUsign [31]) or different trapdoors (e.g. hierarchical identity based encryption [32]). A different class of non-trapdoor signatures over lattices by Lyubashevsky [33] using Fiat-Shamir's heuristic [34] is briefly described as follows:

1. Signer generates a random short vector y and applies it to the lattice obtaining u
2. Signer computes the non-interactive challenge c by computing the hash of u and the message to be signed
3. Signer then finds the proof which is a short vector z by adding y to the product of the secret key (i.e. the short basis and error vector, if LWE is used) and the challenge c.
4. To prevent leakage of secret key information, the signing process may need to be re-run to find a z that is statistically independent.
5. Both z and c make up the signature of the message
6. For verification, the verifier needs to verify that z is a short vector, and uses z and c with the public key (the long basis) and the same hash function to verify the signature.

Of the 3 candidate signature algorithms in the 2nd round of NIST PQC standardization, two candidates (Crystals-Dilithium and qTESLA) are based on Fiat-Shamir signing with LWE while one (Falcon) is based on hash-and-sign with NTRU.

2.1.1.1 Crystals-Dilithium

Crystals (Cryptographic Suite for Algebraic Latties, www.pq-crystals.org) provides both a key-exchange algorithm, Kyber, as well as a digital signature algorithm, Dilithium where both are 2nd round PQC candidates. Crystals-Dilithium uses the Fiat-Shamir signing scheme [34] on a module-LWE problem.

The designers deliberately wanted to reduce the public key and signature sizes [35] by improving on the work by Bai and Galbraith [36] and added the concept of "hints" [37] to do so. To achieve 128-bit security strength, the designers have recommended Dilithium-1280x1024 which has a secret key of 4.8kbytes, public key of 1.4kbytes and signature size of 2.7kbytes. Dilithium takes 250K cycles for key generation, 1000K cycles for signing and 300K cycles for verification which translates to less than 1ms per signature.

2.1.1.2 Falcon

Falcon (Fast-Fourier Lattice-based Compact Signatures over NTRU, www.falcon-sign.info) is based on GPV [30] by applying the SIS problem over NTRU lattices [27]. For the trapdoor sampling, the designers used fast-fourier sampling [38] to improve signing time while achieving a shorter short vector.

For a 128-bit security strength, the designers of Falcon have recommended a Falcon-512 [39] which has a secret key size of 1.2kbytes, public key size of 0.8kbytes and signature size of 0.6kbytes. On a moderate-powered computer, Falcon takes 6.9ms for key generation, and can do over 6k signatures per second and 37k verifications per second.

2.1.1.3 qTESLA

qTESLA (www.qtesla.org) also uses the Fiat-Shamir scheme [34], over a Ring-LWE problem. Similar to Crystals-Dilithium, it enhances on work by Lyubashevsky [33] and Bai-Galbraith [36].

For 128-bit security strength, the designers of qTESLA have recommended qTESLA-p-I which has a secret key size of 5.1kbytes, public key size of 14.8kbytes and signature size of 2.5kbytes. Performance of qTESLA is moderate compared to other lattice-based algorithms with key generation taking 2.3k cycles, signing taking 2.3k cycles and verification at 0.6k cycles. This translates to 0.88ms for a signing + verification flow.

2.1.2 Code-based Cryptography

Code-based cryptography was originally proposed by McEliece [40] in 1978 which described an asymmetric key cryptographic system based on the hardness of decoding a generic linear code, a NP-hard problem [41]. A linear code is essentially a form of error-correcting codes with linear combination properties. The private key in code-based cryptography is typically a code C, which has the ability to correct t errors. When sending a message, the sender will encode the message with the public key and include t errors within the encoding, and the receiver with code C will be able to decode the message while accurately correcting the errors. Typical key sizes for code-based cryptography exceed 1mbits [42] to achieve 128-bit strength in security.

There were 2 digital signature proposals in the NIST PQC Standardization round 1, but both had attacks published in the subsequent public consultation and neither managed to move to round 2. Hence, code-based cryptography is considered only for PQC key exchange, and not for digital signatures under NIST PQC standardization.

2.1.3 Multi-variate Cryptography

Multi-variate cryptography is a broad class of cryptographic techniques encompassing algorithms that rely on the difficulty of solving n unknowns (or variables) within p multivariate polynomial equations. When performing digital signing, the set of p equations (with s unknowns) is the public key while the appropriate values of n is the signature. During signing, the signer has in possession of the private key which consists of 2 affine transformations and a carefully crafted set of polynomial equations that allow for the trapdoor function computation of n from the hash of the message. The verifier can apply n into the p equations to verify that the output of the p equations corresponds to the hash of the message that is signed. While it sounds intuitively NP-hard [43], much of the security lies with the underlying multi-variate scheme, the choice of the parameters s and p, and the design of the trapdoor function needed to support public key cryptography[44].

One of the earliest multi-variate cryptographic constructs was by Matsumoto and Imai [45] who proposed C* in 1988. It was subsequently broken by Patarin [46] who used the general principle to introduce Hidden Field Equations (HFE) [47] and Balanced Oil and Vinegar [48], both broken by Kipnis and Shamir in [49] and [50] respectively. Kipnis et al then introduced Unbalanced Oil and Vinegar (UOV) [51] which is the basis for LUOV and Rainbow, 2 of the 4 multi-variate algorithms in the 2nd round of the NIST PQC standardization. The attractiveness of multi-variate cryptography as a PQC candidate lies in its promise to have a much smaller key size. Sflash [52], a C* variant, was included in European Consortium NESSIE Project [53] in 2003 due to its ability to run in an 8-bit smartcard, but was broken in 2007 [54]. NIST has stated in their report [55] that they hoped to see more security analysis and optimizations in the various multi-variate candidates.

2.1.3.1 GeMSS

GeMSS (A Great Multivariate Short Signature, www-polsys.lip6.fr/Links/NIST/GeMSS.html) is based on HFE [47] and enhances the work from Quartz [56]. As Quartz was designed to generate very small signatures of 128 bits, the designers designed GeMSS as a faster variant of Quartz with better signing efficiency.

For 128-bit security strength, the designers have recommended GeMSS-128 [57] which has a secret key size of 14.2k bytes, public key size of 417.4 kbytes and a signature size of 48 bytes. Performance is a sticking point with GeMSS where key generation takes 44 ms, verification takes 0.041ms but even the most optimized signing process takes over 300ms per signature.

2.1.3.2 LUOV

LUOV (Lifted Unbalanced Oil and Vinegar) is based on UOV, but with an extended output (i.e. the hash of the message to be signed) of the polynomial equations to be another polynomial field, rather than a binary field. The idea is that this will significantly increase the difficulty of the problem which conversely will allow the public key to remain small.

The designers have claimed that for level 2 security (higher than 128-bit security strength), both LUOV-8-63-256 and LUOV-48-49-242 [58] which have secret key size of 23 bytes, public key sizes between 15.5kbytes and 7.3kbytes, and signature sizes between 0.3kbytes and 1.7kbytes respectively. Performance of LUOV-8-63-256 is key generation at 21 million cycles, signing at 5.8 million cycles and verification at 4.9 million cycles which translates to just over 1ms per signature.

2.1.3.3 MQDSS

MQDSS (www.mqdss.org) is based on 5-pass identification scheme [59] while using the Fiat-Shamir transform [34] to arrive at a digital signature construct. It extends on the work by El Yousofi et al [60] by providing a complete and secure construction with proof.

For 128-bit security strength, the designers have recommended MQDSS-31-64 [61] which has a secret key of 64 bytes, public key of 72 bytes and signature size of 39.9kbytes. Performance is key generation at 2.4 million cycles, signing at 14.5 million cycles and verification at 9.6 million cycles which translates to over 3ms per signature.

2.1.3.4 Rainbow

The Rainbow signature scheme was first introduced in 2005 [62] as a generalization of UOV to allow for multiple layers, each with different parameters chosen. Additional layers may improve the security strength of the overall signature construct, but impacts the efficiency and resources needed by the signing and verifying entities.

For 128-bit security strength, the designers have recommended GF(16),32,32,32 [63] which has a secret key size of 93kbytes, public key size of 149 kbytes (which can be compressed to 58.1 kbytes) and a signature size of 64 bytes. Performance is key generation at 35 million cycles, signing at 0.4 million cycles and verification at 0.1 million cycles which translates to 0.12ms for signing.

2.1.4 Hash-based Cryptography

Hash-based digital signature was first introduce by Lamport in 1979 [64]. The concept relies on the difficulty of finding the pre-image of the hash function which is essentially the preimage resistance property [65] in good one-way hash functions. As a start, the signer must first randomly generate a random set of values *s* which is minimally twice the size of the message, as well as the corresponding set of values *p* which are the hash result of the values in *s*. This random set of values *s* is the secret key, and while the set of hashes *p* is the public key. Each 2 values in sequence within sets *s* and *p* relate to 1 bit in the message and signing each bit (either 0 or 1) in the message by the signer involves choosing the appropriate value within the private key set to be published as the signature for that bit. The verifier can then hash the signature and verify the hash against the corresponding public key to ascertain that the correct bit, either 0 or 1, is signed. We can imagine that for large messages, the secret and public key sets, as well as the signature, will become exceedingly large. Also, the keys in the Lamport digital signature are one-time use, and new sets of secret and public keys need to be re-generated for every signature.

There has been several improvements to Lamport's one-time signature (OTS) through variations in the use of Merkle trees [66-69] to extend the function of secret key into a multi-use derivation secret, as well as to reduce the size of the signature. These formed the basis for stateful hash-based signatures which is discussed in Section 2.1.7. On the other hand, to achieve a finite number of stateless signatures, Reyzin's few-time signature HORS (Hash to Obtain Random Subset) [70] transforms an OTS into a N-time signature scheme where the same private key can be used to securely sign N signatures. As each signature reveals a portion of the private key, there is a security degradation from the N+1 signature onwards. SPHINCS by Bernstein et al [71] builds on Goldreich's [72] stateless hyper tree construct to obtain more private signing keys, and uses HORST (adapted from HORS with trees) as the leaves of the trees to increase the number of signatures per key. SPHINCS+, based on SPHINCS, is the only hash-based digital signing candidate in the 2nd round of NIST PQC standardization.

2.1.4.1 SPHINCS+

SPHINCS+ (Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures, www.sphincs.org) is a stateless hash-based signature and improves on SPHINCS' [71] HORST design to obtain faster and small signatures.

For 128-bit security strength, the designers have recommended SPHINCS$^+$-128s [73] as the smallest implementation. The size of the secret key is 64 bytes, public key is 32 bytes and the signature size is 8 kbytes. Performance is key generation at 307.4 million cycles, signing at 4606.9 million cycles and verification at 5.5 million cycles. Signing time has always been the biggest issue faced by hash-based cryptography compared to other signature schemes and this translates to about 1 second per signature.

2.1.5 Isogeny-based Cryptography

Isogeny refers to the mathematical mapping or morphism between 2 mathematical structures. In the case of [74], the public key system is based on the difficulty of finding the isogeny of 2 elliptic curves.

In a mapping over an elliptic curve E where the secret isogeny Φ is mapped to E/<P>, and the secret isogeny ψ is mapped to E/<Q>, revealing E, E/<P> and E/<Q> does not allow the adversary to know Φ or ψ. Hence, similar to a Diffie-Hellman[75] key exchange, two communicating parties can each generate a respective secret Φ and ψ, and arrive at E/<P,Q> to form a shared-secret securely. The most promising advantage of isogeny-based cryptography is the size of the keys which is by far the smallest compared to all the other schemes. As the scheme is based on elliptic curves, key sizes range at 768bits to 1kbits for an equivalent 128-bit strength in security. Unfortunately, no isogeny-based digital signature schemes were submitted for the NIST PQC standardization. Hence, isogeny-based cryptography as with code-based cryptography, is only considered for PQC key-exchange.

2.1.6 Zero Knowledge

Zero-knowledge proofs have their origins in 1985 when Goldwasser, Micali and Rackoff [76] defined the concept of zero-knowledge as proofs that "convey no additional knowledge other than the correctness of the proposition". Since then, many zero-knowledge proof algorithms have been proposed that are broadly proof of knowledge or identity [77] of which the oft-quoted story of Ali-Baba cave by Quisquater et al [78] belongs to; or proof of property, range or set membership [79, 80] which for example allows someone to prove that his/her age is above a certain required value, but reveals nothing else about the age.

Digital signatures are essentially non-interactive proofs of knowledge, but they may not be zero-knowledge proofs. When using an RSA secret key to sign a message, the signature contains information that is algebraically related to the secret key and hence cannot be deemed as zero-knowledge. Zero-knowledge algorithms are typically built on top of a NP problem [81]. To achieve a signing construct, the signer (or prover) needs to generate a signature (or non-interactive proof) to demonstrate knowledge of the secret key (or witness) without revealing any more information to the verifier.

Non-interactive zero-knowledge proofs of knowledge (NIZKPoK) constructions such as MPC-in-the-head [82], ZKBoo [83] and ZK-snarks [84] rely on multiparty computation (MPC) with collision-resistant one-way functions, which could be in the form of strong hash functions (e.g. SHA2-256, SHA3-384) or symmetric key encryption functions (e.g. AES-256) to complete the proof. Informally, the MPC zero-knowledge proof works by the prover splitting the secret into multiple shares (e.g. using exclusive-OR) and committing to the hash of each share. When the verifier challenges the prover on a subset of the shares, the prover is able to produce a "view" for the subset without revealing the actual values. Repeated challenges will increase the assurance that prover has knowledge of the secret.

2.1.6.1 Picnic

Picnic (https://microsoft.github.io/picnic) is NIZKPoK which uses ZKB++ [85], a variant of ZKBoo, as the zero-knowledge proof where the underlying MPC circuit is the LowMC [86] encryption scheme and the hash function used is SHAKE [87] (a SHA-3 derived function). The Picnic signature scheme is made non-interactive through the use of either the Fiat-Shamir [34] or the Unruh [88] transform.

To achieve 128-bit security strength, the designers have recommended picnic-L1-FS [89] which has a secret key size of 16 bytes, public key size of 32 bytes and signature size of 34Kbytes. Performance for key generation is fast as it is simply a random generation operation, followed by a LowMC encryption. Signing time is 1.9ms while verification time is 1.3ms per signature.

2.1.7 Stateful Hash-based Signatures

In the interim while the PQC standardization is taking place, NIST also has a parallel project to work with IETF (Internet Engineering Task Force) to standardize 2 stateful hash-based signatures (HBS) [90], XMSS (eXtended Merkle Signature Scheme) [68] and LMS (Leighton-Micali Signatures) [91] for specific digital signing use-cases. The premise is based on the fact that hashing is provably relatively resistant [13] to quantum cryptanalysis and hence would be a logical choice as an interim solution while the eventual PQC candidate(s) is being chosen. The European Union PQCRYPTO project (http://pqcrypto.eu.org) has also released initial recommendations [17] in 2015 mentioning the use of hash-based signatures XMSS and SPHINCS [71], a stateless hash-based signature and a pre-cursor to SPHINCS+, for use as post-quantum signatures. The advantage that stateful hash-based signatures have over stateless hash-based signatures is the relatively smaller signature size. On the other hand, stateful hash-based signatures require the signer to keep track of key usage or "state" such that all keys are one-time use.

As recognized by NIST and the public comments received [92], stateful hash-based signatures can be deployed in applications where the private key signing is not used often, while public key verification happens more often. The use-cases cited are for code-signing and the issuance of certification authority PKI root-certificates.

2.1.7.1 XMSS

XMSS (eXtended Merkle Signature Scheme) makes use of a Merkle tree of height H to support 2^H W-OTS (Winternitz-One Time Signatures) [67] per secret key. The difference between XMS and other Merkle-tree OTS constructs is the use of random bitmasks in addition to the hash function between the tree nodes. XMSS is provably secure in the standard model, EUF-CMA secure and even forward secure [93].

To achieve 128-bit security strength, the designers have used AES-128 as the pseudo-random function [93] with tree height H=20, which supports 2^{20} ~1,000,000 signatures per secret key. The secret key is 19 bytes, public key size is 912 bytes, and signature size is 2.4kbytes for a 256-bit message. Performance of XMSS key generation is documented to take 109 seconds, but this is because it takes into account the need to generate 1,000,000 secret-public key pairs. Actual signing is 1.7ms and verification is 0.1ms on a high-end laptop. If H=10 is used, then the key supports 2^{10} =1,024 signatures while key generation will only require 0.001 of the time to run.

2.1.7.2 LMS

LMS (Leighton-Micali Signatures) is also based on a Merkle tree structure where the end nodes (or leaves) are W-OTS keys. The main difference for LMS signatures is that each signed message is formatted with a unique identifier header and trailing checksum to prevent collisions.

To achieve 128-bit security strength, LMS uses SHA-256 as the underlying hash function with tree heights supporting H=5, 10, 15, 20 and 25 [69]. At H=20, W=8, the secret key size is 32 bytes, the public key size is 768 bytes and signature size is 3.4kbytes. Performance of LMS is comparable with XMSS with key generation taking 180 seconds for H=20 and 6 seconds for H=15. Signing time and verification time are likely to be similar to XMSS.

2.2 Other Related Work

Beyond the standardization efforts by NIST which we see as comprehensive as it gets to arriving at a possible new quantum-resistant signing algorithm, we consider two other categories of signatures which are not within the scope of NIST PQC efforts. One would be to continue to use RSA / ECC but with increased key sizes and the other would be to use quantum computers to enhance security to defend against adversaries with quantum computers.

2.2.1 Classical Cryptography

The underlying principle to continue to use the familiar RSA and ECDSA, albeit with a larger key size, in the post quantum era is not difficult to imagine. The difficulty, from a practicality standpoint, is that this principle may lead to large and unwieldy key sizes that may not be supported or usable, and that it seems to only delay the inevitable.

We need to first understand how Shor’s algorithm works conceptually, and then arrive at a means of computing the required key sizes. In order to cryptanalyze an RSA public key, we want to find the 2 prime factors that make up the modulus *n*. The factorization algorithm is as follows:

1. If the modulus is even, then one of the factors is 2 (solved).
2. If the modulus is odd, then loop the following:
    a. Generate a random $a < n$.
    b. Find g = GCD(a,n). if $g > 1$, then we have found g as one of the factors. (solved)
    c. Find r = order of a within n. This means $a^r \equiv 1 \bmod n$
    d. If r is odd, go back to step 2.a
    e. If r is even, compute $x = a^{\frac{r}{2}} \bmod n$. If $1 < x < n\text{-}1$, the go to step 3. Else go back to step 2.a
3. Compute p = GCD($x+1$,n) and q = GCD($x-1$,n). p and q are the factors of n.

Where Shor’s algorithm [8] comes into play is only in step 2.c where no classical or probabilistic polynomial-time algorithm exists for the order finding problem [94]. Shor’s algorithm, through the use of phase estimation and quantum fourier transform (QFT), provides O(log N) time complexity to find the order. The number of qubits needed on a quantum computer to break RSA is estimated at 2n+3 [95] and 2n+2 [96] which means that there needs to be a 4,000+ qubit Quantum computer to break an RSA-2048 signature. Shor’s QFT algorithm can also be adapted to solve the discrete logarithm problem and the number of qubits to break ECDSA is “roughly” 6n [9], or a 1,500+ qubits Quantum computer to break a ECC P256 signature.

Post quantum RSA was studied by Bernstein [97] who showed the technical feasibility of implementing a terabyte key using 2^{31} 4096-bit primes as factors. With those key sizes, each RSA operation amounted to tens or hundreds of hours. Whilst there is no examination of the secure lifetime of such large RSA keys, we expect that quantum technology will catch up with these keys in a matter of time. Separately, the study of post quantum ECC is largely towards isogenies which is mentioned in Section 2.1.5.

2.2.2 Quantum Enhanced Security

Quantum cryptography refers to the use of quantum computers to perform certain cryptographic operations. Such operations typically exploit the quantum properties of superposition, interference and entanglement which are not replicable by classical computers. Quantum enhanced security [18] is then the augmenting of classical non-quantum systems to make use of, or be supplemented with, quantum technology to enhance their ability to secure their data and transactions against adversaries who may be fully quantum capable. The mechanism of augmenting could be through the use of blind quantum computing [98] which then allows the quantum computer to perform the protocol on-behalf of the communicating parties without revealing information to the quantum computer.

While quantum key distribution (QKD) [99, 100] has been synonymously equated with quantum cryptography, it is based on the Vernam one-time pad and hence more applicable for key exchange and encryption. Quantum researchers have introduced various quantum digital signature schemes [101-104] but as they typically rely on QKD with entanglement, these signature schemes should be more appropriately labelled data authentication schemes. We were unable to locate any quantum digital signature that possesses the necessary constructs of a digital signature scheme as defined in Section 3.1 and is EUF-CMA (existential unforgeable under chosen message attack) secure, let alone post-quantum secure.

3 Modelling the Signature Scheme

A digital signature is a cryptographic primitive that can be used to achieve data integrity, user authenticity and transaction non-repudiation. Between 2 honest communicating parties Alice and Bob, having

- *Data Integrity* ensures that any message sent from Alice to Bob can be verified by Bob if the message received was unmodified, or had been modified in transit.
- *User Authenticity* ensures that Bob is able to ascertain if Alice is who she claims she is.
- *Non-repudiation* ensures that Bob is able to prove that the message is received from Alice, and Alice is unable to deny that proof.

While data integrity and user authentication can also be achieved using other security primitives such as message authentication codes (MAC), only digital signatures can provide the non-repudiation capability that is needed in many business applications.

3.1 Digital Signature Scheme

We define a digital signature scheme DSS = (KeyGen, Sign, Ver) as a triple of polynomial time functions with the following parameters:

KeyGen(1^n) $\rightarrow (K_s, K_p)$ takes in a security parameter 1^n which typically defines the cryptographic key strength of n, and outputs a secret key K_s and corresponding public key K_p.

Sign(M, K_s) $\rightarrow (\sigma)$ takes in a message M and the secret key K_s, and outputs a signature σ.

Ver(M, K_p, σ) $\rightarrow$ (result) takes in a message M, the public key K_p and signature σ, and outputs *accept* if and only if σ is a valid signature generated by Sign(M, K_s).

KeyGen() and Sign() are functions performed by the signing party, while Ver() is the function performed by the verifying party.

We next classify the DSS as secure if it is proven to be existential unforgeable under chosen message attack (EUF-CMA) [105]. The EUF-CMA experiment between Alice and adversary Mallory seeks to prove that Mallory will not be any closer to forging a signature despite multiple interactions with Alice. It goes as follows:

1. Alice performs KeyGen() and sends the public key K_p to Mallory.

2. Mallory can choose a message M_i and ask Alice to sign the message.
3. Alice signs the message M_i using Sign(M_i,K_s) and returns the signature σ_i to Mallory. Step 2 and 3 can be repeated multiple times.
4. At the end of the experiment, Mallory has to output a message M' that is not within the set of messages M_i requested in Step 2, and also a signature σ' that will return *accept* when Ver(M', σ',K_p) is called.

There are classifications for stronger DSS requirements such as strong existential unforgeable under chosen message attack (SUF-CMA) which has an additional unmalleability [106] requirement. But for NIST PQC evaluation criteria [19] and purposes of this study, a DSS that is EUF-CMA secure is sufficient. We note that the basic RSA signature algorithm is not EUF-CMA secure. It needs to be used in a PKCS#1 v1.5 or full-domain hash (RSA-FDH) in order to be EUF-CMA [107, 108] secure.

3.2 Signature Parameters

With the basic understanding of what makes a DSS, we can next identify the different characteristics that vary depending on the signature scheme. NIST in its call for proposals [19] had listed 3 broad categories, namely security, cost and simplicity, as their evaluation criteria. Within the cost criterion, specific parameters include size of keys, computational efficiency of operations and possible decryption failures. The intention in this section is not to repeat the evaluation, but to establish a comparable feasibility basis between what a signature scheme requires versus what the business application can support, on the assumption that the signature scheme remains secure. For example, if a signature scheme requires key sizes of over 10kbits in size to be secure, but the business application can only support key sizes of 2 kbits, then the signature scheme is not applicable for the business application, no matter of the security or operational efficiency. For each of the parameters, we define 3 ranges for small, medium and large and attempt to provide suitable ranges for classifying each of the signature scheme requirements into one of these ranges.

3.2.1 Key Generation

Key generation is by far the most sensitive operation for any DSS. If not done correctly, it leads to systemic vulnerabilities [109] which are extremely costly to fix. The parameters to be classified in key generation are:

3.2.1.1 KeyGen() Resources

Key generation is typically done by the signing party and may require access to specific resources such as a random number generator (RNG), as well as sufficient memory / compute power to verify the validity of the key (e.g. probabilistic primes. We classify 3 ranges for "KeyGen() Resources" as:

KeyGen() Resources	**Small (Optimal)**	**Medium**	**Large**
	Able to run on a chip card.	Able to run on a terminal / mobile phone	Requires to run on a server or a cryptographic processor (e.g. HSM)

3.2.1.2 Key Size

Once the secret key K_s and corresponding public key K_p are generated, the secret key needs to securely stored within the signing party, while the public key needs to be transmitted or disseminated to the verifying party. We classify 3 ranges for "Key Size" as:

Secret Key Size, Public Key Size	**Small (Optimal)**	**Medium**	**Large**
	< 2Kbits (e.g. ECC-P256)	< 2Kbytes (e.g. RSA-8192)	> 2Kbytes

3.2.1.3 Key Lifetime

Certain DSS may have a limitation of how many signatures can be supported for the specific key. If a secret key has a short lifetime, then the signing party will require to go through the KeyGen() function and disseminate the public key frequently. We classify 3 ranges for "Key Lifetime" as:

Key Lifetime	**Small**	**Medium**	**Large (Optimal)**
	Limited. (< 1000 signatures)	Limited. (< 10000 signatures)	Unlimited

3.2.2 Signing

3.2.2.1 Sign() Resources

The signing party, with possession of the secret key K_s, needs to carry out the Sign() function to generate the signature σ. We classify 3 ranges for "Sign() Resources" as:

Sign() Resources	**Small (Optimal)**	**Medium**	**Large**
	Able to run on a chip card	Able to run on a terminal / mobile phone	Requires to run on a server or a cryptographic processor (e.g. HSM)

3.2.2.2 Signature Size

The output signature σ generated by the signing party needs to be transmitted to the verifying party and possibly be stored by the verifying party as proof of non-repudiation. A small signature size will be much more efficient in transmission and storage as compared to a large signature size. We classify 3 ranges for "Signature Size" as:

Signature Size	**Small (Optimal)**	**Medium**	**Large**
	< 10 kbits (e.g. ECC-P256)	< 10Kbytes (e.g. RSA-8192)	> 10Kbytes

3.2.2.3 Signature Time

The time needed for the signing party to perform the Sign() function is critical for several business applications. Assume that the signing function is running on a high-end laptop, we classify 3 ranges for "Signature Time" as:

Signature Time	Small (Optimal)	Medium	Large
	< 1 ms per signature	< 100ms per signature	> 100 ms per signature

3.2.3 Verifying

3.2.3.1 Ver() Resources

The verifying party, upon receiving the message M and signature σ, requires to call the Ver() function with the signing party's public key K_p, to check the validity of the signature and authenticity of the message. This also has direct impact on the time needed to perform the Ver() function. We classify 3 ranges for "Ver() Resources" as:

Ver() Resources	**Small (Optimal)**	**Medium**	**Large**
	Able to run on a chip card	Able to run on a terminal / mobile phone	Requires to run on a server or a cryptographic processor (e.g. HSM)

3.2.3.2 Signature Lifetime

This signature lifetime is a business application requirement for the verifier to store the received message and signature for subsequent proof of non-repudiation. We evaluate the DSS's ability to support a long signature lifetime by understanding the storage requirements related to the signature and public key. We classify 3 ranges for "Signature Lifetime" as:

Signature Lifetime	**Small**	**Medium**	**Large (Optimal)**
	< 1 day	< 1 year	> 1 year

3.2.4 Profiling the Algorithms

in Table 2, we profile each of the PQ algorithms described in Section 2 using the signature parameters

Table 2. Parameterized Profile of PQ Algorithms

	KeyGen Resource	**Secret Key Size**	**Public Key Size**	**Key Lifetime**	**Sig Resource**	**Sig Size**	**Sig Time**	**Ver Resource**	**Sig Lifetime**
Dilithium	M	L	M	L	M	M	S	M	L

Falcon	M	M	M	L	S	S	S	S	L
qTESLA	M	L	L	L	M	M	S	M	L
GeMSS	L	L	L	L	L	S	L	M	L
LUOV	L	L	L	L	M	S	L	L	L
MQDSS	M	S	S	L	L	L	L	L	L
Rainbow	L	L	L	L	M	S	S	M	L
SPHINCS+	L	S	S	M	L	M	L	M	L
Picnic	S	S	S	L	M	L	M	M	L
XMSS/LMS H=20	L	S	S	M	M	M	M	S	L
XMSS/LMS H=10	S	S	S	S	S	M	M	S	L
Large RSA Key	L	L	L	L	L	L	L	M	M

4 Applications

In this section, we examine a broad spectrum of 11 different applications across 4 broad categories (see Figure 2). Since NIST is not accepting any more submissions, we can assume that in the best case, all the algorithms are secure and accepted. Hence will we be able to find or choose the best fit algorithm for each of the applications? Or will there be technological gaps, despite the PQC standardization efforts?

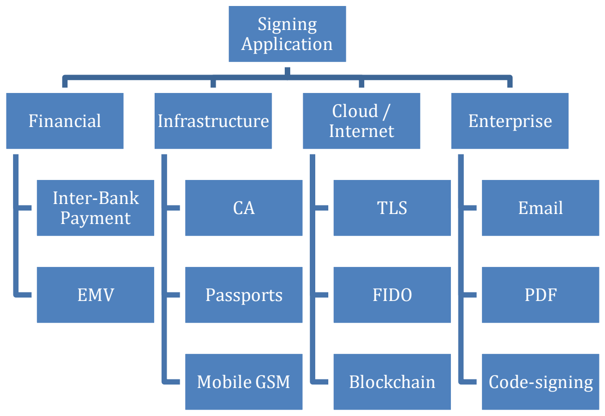

Figure 2. 11 applications that use digital signatures

For each application, we will run through the same exercise in Section 3 on the digital signature parameters but this time from a requirements perspective, that is, what is the minimal range allowed for by each application. The assumption taken here is that we are evaluating the application "as-is", without evaluating the feasibility of the application being modified or adapted to suit a PQ algorithm.

4.1 Financial

Banking and financial institutions require to maintain their reputation as trustworthy organizations before customers are willing to transact with them. The need for security is heavily emphasized both at regulatory levels as well as at industry best practices. We identify 2 applications that use digital signatures for their authentication and transaction non-repudiation needs: inter-bank payment systems and EMV cards.

4.1.1 Inter-bank Payment Systems

On any given working day, banks transfer hundreds of million or even billion dollars' worth of money electronically amongst each other. These transactions could be for trade settlement, currency exchange, loans, borrowings, etc, and they are typically carried out on an inter-bank payment network such as SWIFT (Society for Worldwide Interbank Funds Transfer). Banks have specialized systems and terminals to connect into the

SWIFT proprietary network to send and receive payment instructions where mutual authentication and encryption are provided for. Non-repudiation of messages originating from the banks' customers is supported in SWIFT through their 3SKey offering [110] where corporates and end-users will individually own a 3SKey USB token to participate in a public key infrastructure (PKI) operated by SWIFT.

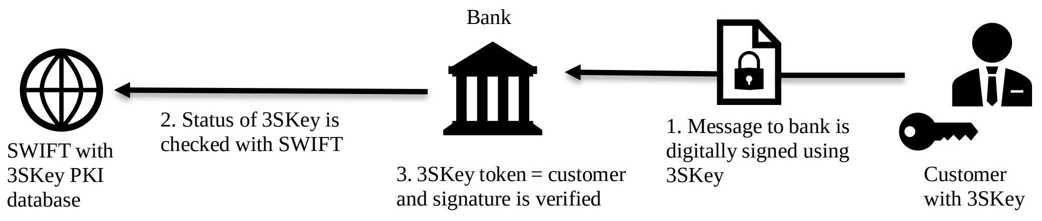
Bank
SWIFT with 3SKey PKI database
2. Status of 3SKey is checked with SWIFT
3. 3SKey token = customer and signature is verified
1. Message to bank is digitally signed using 3SKey
Customer with 3SKey

Figure 3. How 3SKey is used for digital signing messages on the SWIFT network [112]

According to SWIFT's documents, the digital signing algorithm used in 3SKey is RSA-2048 [111]. The digital signing process (see Figure 3) is as follows:

1. Messages to be sent by the end-user or corporates will be digitally signed using a 3SKey USB token before they are received by the bank.
2. The bank verifies against the SWIFT 3SKey portal that the token has not yet been revoked.
3. The bank further verifies that the sender is indeed issued with the token that was used to sign the message.
4. If the signature is verified OK and the above 2 checks are done, then the digital signature from the user is deemed valid.

We proceed to map the 3SKey's signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
3SKey	S	M	M	M	S	M	M	L	L

Most importantly, we expect to see the availability of a USB token that can support quantum-resistant algorithms before this solution can be migrated over.

4.1.2 EMV Cards

EMV (Europay-Mastercard-VISA) is chip-based standard that is used for consumer payments. Consumers are provided with a chip-card that is EMV-compliant and can be used to pay for purchases at physical merchant stores, such as restaurants, boutiques and service parlors. Merchants have specialized terminals that are able to read the EMV chip-cards and obtain the necessary authentication and transaction authorization from the issuing banks via the payment networks. While majority of the crypto algorithms used in EMV is symmetric-key based, EMV relies on asymmetric key digital signatures in the static data authentication (SDA), dynamic data authentication (DDA), and combined data authentication (CDA) protocols [3] to reduce payment card fraud. In 2017 alone, there is an increase of over 1 billion EMV cards in circulation over the previous 12 months [112]. EMV cards expire every 3-5 years where they are replaced with new keys.

SDA is the entry-level EMV mechanism to deter fraudsters from illegally generating their own card data to fool merchants. It achieves this by requiring the issuing bank to digitally sign the card information during EMV card personalization and include the signature statically within the card. When the consumer presents the card to the merchant, the terminal is able to verify the digital signature before processing the transaction, and the final application cryptogram can then be verified by the issuing bank via the payment network. While SDA is a significant step-up from magnetic stripe payment cards, SDA has a drawback due to the static nature of the signature. It does not prevent offline card cloning fraud where fraudsters copy public information from a valid card, including the signature, into another card and attempt to perform an offline payment transaction.

This means that merchants or transaction scenarios that require offline EMV transactions (e.g. when network connectivity is unstable or intermittent, or kiosks in remote locations) are still vulnerable to card fraud.

DDA is often recommended as the enhanced and more secure EMV mechanism for card fraud prevention. It includes all the features of SDA, with an added functionality on the EMV card to digitally sign a challenge from the merchant terminal. While this added functionality brings up the cost of an EMV DDA card versus an EMV SDA card, the main advantages of DDA over SDA are that it defeats offline card cloning fraud and also supports encrypted pin entry between the merchant terminal and EMV card. CDA is a variant of DDA where the EMV card signs both the terminal challenge as well as the application cryptogram. Figure 4 shows the key management lifecycle for an EMV DDA card.

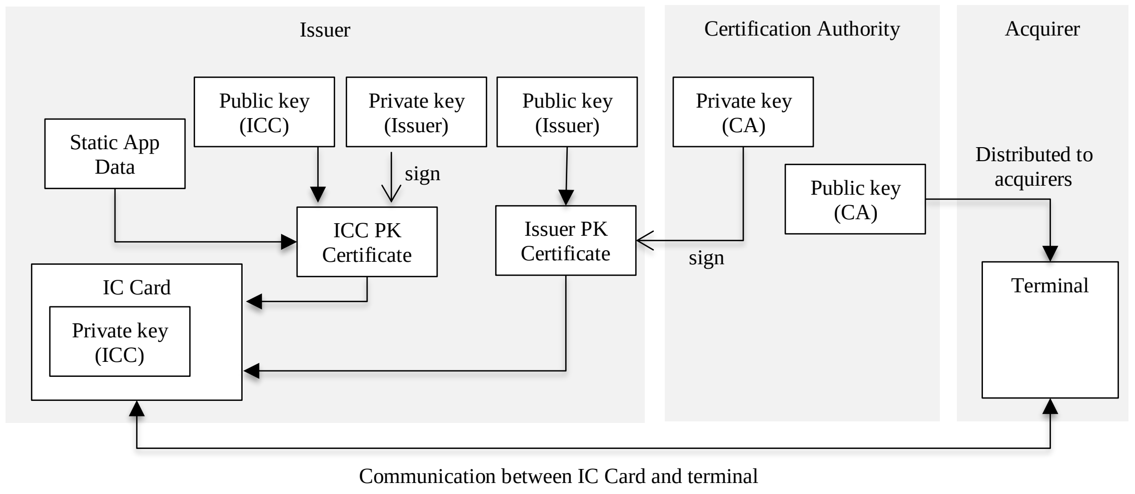

Figure 4. DDA under EMV [3]

Within the DDA protocol, there are 3 signing keys used:

- The Certification Authority (CA) key is used to issue a certificate by signing on the Issuer public key. The CA public key is embedded in the merchant terminals as the root-of-trust.
- The Issuer (I) key is generated in the issuing bank's infrastructure, typically in a Hardware Security Module (HSM) and is used to sign card personalization information to be injected into the EMV Card. The certificate of the Issuer public key is also injected into the EMV card. This process is similar for both SDA and DDA.
- The Integrated Circuit Card (IC) key is generated internally within the EMV card, where the public key is exported and signed by the Issuer key. This key is used for signing the challenge during the EMV Card's communication with the merchant terminal for authentication.

The signing algorithm to be used for EMV is RSA of up to 248 bytes (1984 bits), while the hashing algorithm used is SHA-1 [3]. The reason for the 248-byte limit is due to the communication interface standard (ISO-7816) between the EMV Card and merchant terminal. Based on ISO7816-4 [113] protocol specification, the maximum size of a message is 255 bytes, and if we deduct the 7-byte header and footer, it leaves only a maximum of 248 bytes as the maximum signature size that can be transmitted between the EMV Card and terminal in a single message exchange. We proceed to map the EMV signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
EMV (SDA)	L	L	S	L	L	S	L	M	L

EMV (DDA)	S	S	S	M	S	S	S	M	S

The largest constraint in EMV Cards has to do with the ISO7816 protocol which unfortunately restricts each message to 255 bytes. If a larger signature size is needed, then the standard allows for extended APDU formats (increasing the total transmission size to 65536 bytes) but this will need to be supported by both the card and the terminals.

4.2 Infrastructure

The security of public infrastructure affects the residents that rely on the infrastructure for their day-to-day operations.

4.2.1 Certification Authority

Certification Authorities (CA) play a key role in providing the root-of-trust or trusted third party between different entities. A CA can issue digitally signed certificates attesting to the identity, public keys and other attributes of an entity electronically such that other relying parties wanting to communicate or transact with this said entity can rely on information in the certificate as reliably as the trust given to the CA issuing the certificate (see Figure 5). ITU (International Telecommunication Union) maintains the X.509 [1] framework as the globally recognized standard for certificates issued by CAs, and also covers the necessary ancillary services needed to maintain the security, inter-operability and availability of certificates including directory services, policies, hierarchy, cross-certification, etc.

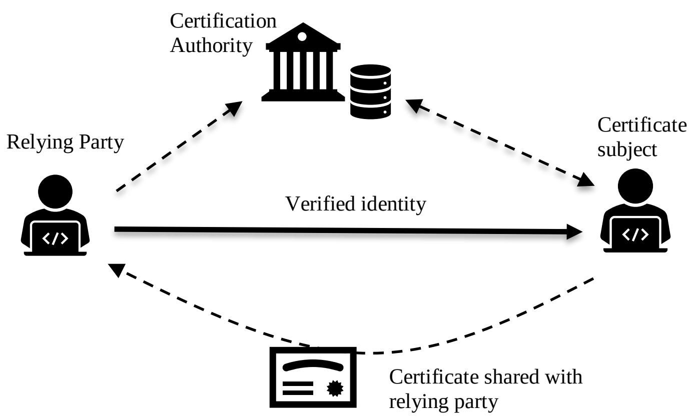

Figure 5. Trusted third party role by CA [1]

Because of the extendible structure of X.509, it can accommodate new algorithms that are being added to the community. IETF has already started on a draft to assign an algorithm identifier for XMSS [68, 114], and this does not require any change for X.509. At the top of a certificate hierarchy, CAs issue a root CA certificate which is a self-signed certificate to be trusted by communicating entities. Below the root certificate, CAs may issue intermediate CA certificates for availability or segregation of policy purposes before the actual user certificates are issued. Should the intermediate or user certificates are compromised or lost, these certificates can be revoked by including them in the certificate revocation list (CRL) signed by the root CA in a timely manner to inform all relying parties. However, if the root CA certificate is compromised, then the impact and fallout become significant and companies may become bankrupt (such as in the wake of the DigiNotar attack [115]). Most CAs protect the root secret key in HSMs, behind layers of firewalls and physical separation, but this is insufficient when powerful quantum computers exist.

We proceed to map the CA key requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
CA Key	L	L	M	M	L	L	L	M	L

4.2.2 Passport / IDs

ICAO (International Civil Aviation Organization) maintains the ICAO 9303 [116] standard for MRTD (machine readable travel documents) or e-Passports which is the officially recognized standard used today for identifying persons in all cross-border travel. The e-Passport contains a data page which includes both visual information as well as electronic information (including personal particulars, facial & fingerprint biometric, visa, authentication keys) embedded within the contactless-IC chip about the person identified by the document.

The electronic information in the e-Passport is signed by a document signer key which is certified by the country signing certification authority (CSCA). CSCA are root CAs whose self-signed certificates are uploaded into the ICAO public key directory for dissemination to other countries. Document signer keys are short-term usage keys to sign the electronic information but need to remain secure for the entire lifetime (5-10 years) of the e-Passport. We proceed to map the ICAO Document signer signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
ICAO 9303 Document signer	L	L	S	S	L	S	L	M	L

In order to prevent substitution of the contactless-IC chip in e-Passports, ICAO 9303 also introduces the concept of active authentication and/or chip authentication. In both these authentication flows, the contactless-IC chip is required to digitally sign a challenge or perform a key exchange with the inspection system in order to prove that the chip is authentic. Authentication algorithms defined for active authentication (RSA and ECDSA) or chip authentication (DH and ECDH) do not include post quantum algorithms. As the requirement is very similar to the EMV-DDA scheme, we do not need to include the requirements in this evaluation.

4.2.3 GSM eSIM

GSMA (Global System for Mobile communications Association) is a trade grouping of technology vendors, network operators and service providers that define, implement and certify standards used for modern day mobile communication using handsets. Much of the security related to mobile communications use industry best practices including AES-128 bit encryption, SHA-256 hash and key derivation, and common-criteria assurance level for SIM (subscriber identity module) cards.

As the mobile industry moves towards eSIM [117] where subscriber profiles can be dynamically provisioned over the air (OTA) into the secure element, digital signatures are used as part of the GSMA PKI to issue X.509 certificates to the various communicating entities to establish the chain of trust from the point the data is prepared, via the mobile network operation, into the eUICC (universal integrated circuit card). Embedded within the eUICC are several keys and certificates including SK.EUICC.ECDSA (eUICC’s secret key for signing), CERT.EUICC.ECDSA (eUICC’s certificate of the public key), PK.CI.ECDSA (the PKI root certificate), all of which can only be NIST P256, brainpoolP256r1 or FRP256V1 elliptic curves [118]. During provisioning, a common flow is as follows:

1. MNO (mobile network operator) requests SM-DP (subscriber management data preparation) system to prepare a profile for the subscriber.
2. Subscriber uses the handset’s LPA (local profile assistant) to scan or input an activation code to connect to the appropriate SM-DP.
3. LPA enables eUICC and SM-DP to perform mutual authentication:
    a. eUICC generates a eUICC-challenge which is sent to the SM-DP.
    b. SM-DP digitally signs the SM-DP response which includes SM-DP address, transaction ID, server-challenge and eUICC-challenge.
    c. eUICC verifies the SM-DP signature, and digitally signs the eUICC response which includes the transaction ID, server-challenge and relevant eUICC information.
    d. SM-DP verifies the eUICC signature.
4. SM-DP signs a confirmation request to prompt the subscriber to confirm the profile installation.
5. Subscriber’s profile is then installed into the eUICC.

6. eUICC and SM-DP will perform a key agreement using elliptic curve Diffie-Hellman (ECDH) to establish a session key.
7. Encrypted profile is downloaded by the LPA, and cut into small sizes to be loaded into the eUICC.
8. SM-DP receives notification that installation is successful.

We proceed to map the eSIM signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
GSM eSIM	L	S	M	S	S	M	M	S	S

We note that eUICC, an EAL 4+ compliant secure element, is required to perform much of the key operations to complete the authentication process.

4.3 Cloud & Internet

The Internet and cloud have seen the highest growth in types of applications, volume of transactions and number of users. It has also fueled the demand for good security solutions to be used to protect the users, transactions and data.

4.3.1 Transport Layer Security (TLS)

Transport Layer Security (TLS) [2, 23] or its predecessor Secure-Sockets Layer (SSL) is the most commonly deployed transport level security for cloud, Internet, mobile and even IoT applications. Within the OSI 7-layer network model, TLS operates just above layer 3 – Network, and provides add-on encryption, authentication and non-repudiation functionality for protocols such as Border Gateway Protocol (BGP), Domain Name Service (DNS), Hyper-text Transfer Protocol (HTTP), Telnet, File-Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and many more. HTTPS (HTTP over SSL) is the TLS-secured version of HTTP, and is the de-facto protocol used for transferring web-page content between the organization web-servers and the end-users' web-browser. Similarly, DNSSEC (DNS Security Extensions) is a TLS-secured version of DNS to ensure clients can rely on signed DNS responses from valid DNS servers.

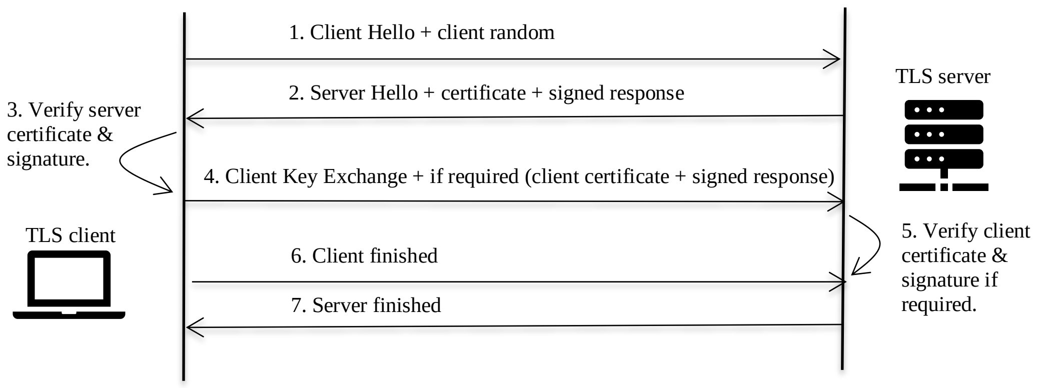
1. Client Hello + client random
2. Server Hello + certificate + signed response
TLS server
3. Verify server certificate & signature.
4. Client Key Exchange + if required (client certificate + signed response)
TLS client
5. Verify client certificate & signature if required.
6. Client finished
7. Server finished

Figure 6. Overview of TLS v1.2 handshake [119]

The TLS v1.2 mechanism to protect these protocols lies in the use of cryptographic primitives deployed in a multi-way protocol called a TLS handshake [119, 120]. The end-goal of the TLS v1.2 handshake is for client and server to authenticate the identities of the other party, and also establish a shared secret key that can be used for encrypting subsequent messages exchanged between the communicating parties. After the standard 3-way TCP handshake (SYN-SYNACK-ACK), the high-level TLS v1.2 handshake protocol (see Figure 6) happens as follows:

1. The client issues a “Client-Hello” message with the list of cipher-suite and other cryptographic information supported, in order of preference, as well as a client-random to be used to as part of the session key-exchange.
2. The server chooses the most appropriate cipher-suite based on the intersection and its own preference choices, and responds a “Server-Hello” to the client with the chosen cipher-suite, as well as the server certificate which contains a signed copy of the server’s public key by a certification authority (CA) acceptable by the client, and a signed response containing the client-random, server-random and other information to complete the session key-exchange. The server may also optionally request for the client certificate for mutual authentication over TLS. If no client certificate is requested, then it is assumed that the server has other means (at application level) to verify the identity of the client.
3. The client, upon receiving the server certificate, will verify the signature of the client-random followed by the certification of the server’s public key, through the chain of trust, to a CA recognized by the client. If verified ok, this means that server authentication as well as client session key exchange is complete.
4. The client responds to the server with additional cryptographic information for the server to complete its session key exchange, and also, if requested, with the client signature and certificate to prove its identity.
5. The server, upon receiving the cryptographic information from the client, will complete the session key exchange. It can also authenticate the client through the client signature and certificate if mutual authentication over TLS is required.
6. Session key exchange is now complete, and the 2 parties can start to exchange information in an encrypted and authenticated (either 1-way server authentication, or 2-way mutual authenticated) session.

Throughout the TLS v1.2 handshake, asymmetric key cryptography is used for both the establishment of the session key, as well as for the authentication of the parties. The TLS standard supports a broad spectrum of algorithms including RSA, DSS, ECDSA for signing, and RSA, DH, ECDH for key exchange.

In the TLS v1.3 handshake, the protocol is optimized for speed, achieves perfect forward secrecy (PFS) [22, 121] using Ephemeral Diffie Hellman (EDH) and removing RSA for use in key exchange, and adds in additional signing algorithms such as EdDSA. No post-quantum algorithms have yet been added, but the full forward secrecy design for key exchange means that systems that implement TLS v1.3 will have the confidentiality of the messages protected against subsequent post-quantum cryptanalysis. Since digital signing is used in TLS for authentication, the requirement for long-term signatures is not crucial. We proceed to map the TLS signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
TLS (server)	L	L	L	L	L	L	S	M	S
TLS (client)	M	L	L	L	M	L	M	L	S

We should take note of device considerations on smaller TLS clients such as IoT devices which require to minimally be able to verify the signature from the server during a TLS-handshake for a 1-way server authentication.

4.3.2 Blockchain

The rise of blockchain started with Nakamoto’s Bitcoin white paper [122] in 2008. Since then, we have witnessed a meteoric rise in both resources used in [123] and variations of blockchains, mainly in the area of crypto-currencies. The estimated total value of crypto-currencies fluctuates around US$250 billion currently, down from the peaks of US$500 billion just 3 years ago. This is a very sizable value in money market terms since the tradable value is almost equal to 8-10% of total US dollars in circulation.

The basic Bitcoin architecture is a distributed system of peer nodes which store and maintain the chain of all transaction blocks since the genesis block generated by Nakamoto in 2008. Nodes which choose to be miners can also participate in the chain-growth process to add a new block every 10 minutes by solving a cryptographic puzzle in proof-of-work (PoW) consensus protocol and earning some Bitcoins in the process. Each Bitcoin transaction block contains a collection of signed transactions, collated into a Merkle tree structure where a miner node will verify the validity of individual transactions, including the available balance for spending and the digital signature used to sign the transaction. The algorithm used for signing transactions is ECDSA secp256k1 [124] but with additional extensions [125, 126] to prevent the exposure of the public key until the private key is used the first time to sign a transaction.

As a cryptocurrency, quantities of Bitcoins which are earned through the mining process are logically tagged to wallet addresses. These Bitcoins can be spent or transferred to another wallet address by a transaction which is signed by an ECC private key. The association of the private key to the wallet address is by the hash of the public key whereby a signature for a transaction is valid if it is carried out on the wallet address equivalent to the hash of the public key. In Figure 7, the address to which the payor needs to transfer the bitcoins to is the hash of the payee's public key. This is the Pay-to-PubKey Hash (P2PKH) mechanism which accounts for majority of Bitcoin transactions. Other mechanisms include Pay-to-Script Hash (P2SH) which can be used to support a multi-signature transaction and uses the hash of the script as the wallet address. Hence given any private key, the Bitcoin client will be able to find the wallet address. However, given only a wallet address, no client (even a quantum-capable client) will be able to know the associated public key and hence not be able to cryptanalyze the public key to obtain the private key.

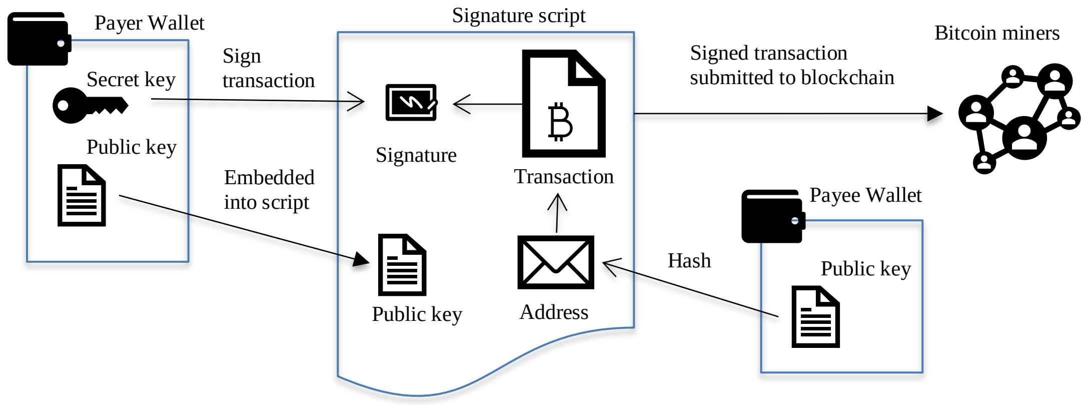

Figure 7. How a P2PKH bitcoin transaction is signed

Note that once the first transaction of the wallet address is created, the public key becomes known throughout the chain and there is no further protection of the public key. BIP0032 [125] proposes the use of a hierarchical deterministic (HD) wallet structure where a master seed is used to derive subsequent one-time use private keys. Using a HD wallet means that Bitcoins are spent and transferred to an entirely new wallet address each and every time a transaction is made. And this means the public keys can remain hidden throughout the lifetime of the master seed. However, Chalkias [127] highlights numerous situations and pitfalls where the public key may be exposed prior to a valid transaction despite best efforts including posted transactions that are not included in blocks, invalid transactions due to insufficient Bitcoin balance, forked blockchains, etc, and hence effectively does away with hiding public key as a strategy for post quantum resistance.

Beyond transaction signing, there are variant blockchain implementations that rely on digital signatures in the consensus protocol such as proof-of-stake (PoS) consensus found in Ouroboros [128] and Dfinity [129]. In this case, chain-growth relies on the block proposer to digitally sign the proposed transaction block to be verified by other nodes in the blockchain. We proceed to map the Bitcoin signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
Bitcoin	M	L	L	L	M	M	M	M	L

4.3.3 FIDO Authentication

In a bid to reduce the reliance of passwords as the primary means of user authentication, the FIDO Alliance (fast identity online) proposed the UAF (Universal Authentication Framework) standard [130] that replaces the secret password that a user needs to enter with a front-end UAF client that could biometrically verify the user and then rely on public key cryptography to sign the authentication challenge to the authenticating server. Organizations that openly recognize FIDO include Google and Microsoft that have included the support for FIDO as part of their product and service offerings.

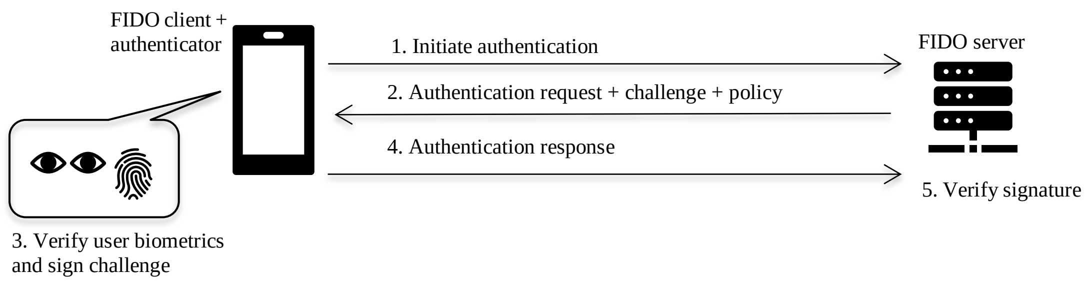
FIDO client +
authenticator
1. Initiate authentication
FIDO server
2. Authentication request + challenge + policy
4. Authentication response
5. Verify signature
3. Verify user biometrics
and sign challenge

Figure 8. FIDO UAF authentication [130]

Within the UAF specifications (see Figure 8), user authentication happens as follows:

1. User on the end-user device (typically a smart phone) initiates the authentication process.
2. FIDO server responds with an authentication request and challenge.
3. FIDO client will activate the relevant FIDO authenticator module on the end-user device to carry out user authentication according to the server policy. This could be in the form of facial or fingerprint recognition for biometrics. If verified ok, FIDO authenticator will sign the server challenge using the user private key. Algorithms supported are ECDSA, RSA and China's SM2.
4. FIDO client sends the signed challenge back to the FIDO server as the proof of authentication.
5. FIDO server can verify the signature to authenticate the user.

We proceed to map the FIDO client requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
FIDO	M	L	L	M	M	L	M	L	S

4.4 Enterprise

Digital signatures are also used within the Enterprise to provide for authenticity and integrity of data and applications.

4.4.1 Email

Pretty Good Privacy (PGP) [131] is a popular open-source email security add-on that is used by millions of users for protecting the confidentiality and integrity of emails being sent. Communicating parties share their asymmetric public keys amongst each other, and then use private key to achieve key exchange and/or non-repudiation [132]. At the high level, for email confidentiality,

1. Sender generates a random value, n, that is used to encrypt the message using a symmetric key algorithm such as 3DES, AES or Twofish.

2. The random value n is then encrypted using the receiver(s) public key(s). The algorithms supported are RSA, ElGamal.
3. Both the encrypted message and encrypted random value n are sent to the receiver(s).
4. Receiver(s) can use their private key to decrypt the encrypted random n before using n to decrypt the encrypted message.

For authentication and non-repudiation,

1. Sender performs a RIPE-MD160/SHA-1/SHA2 hash on the message to be sent.
2. Sender performs a decrypt or signing operation on the hash using the sender's private key. The algorithms supported are RSA, DSA and ECDSA [133].
3. The output of the signing operation is the signature which is sent along with the message which can also be encrypted.
4. Receiver(s) upon receiving the message can perform the same hash, and use the sender's public key to encrypt and verify that the signature corresponds to the hash of the message.

Private key storage and protection for PGP is handled by the PGP client. Most implementations of PGP clients are run on laptops and personal computers, and mobile versions (e.g. www.pgpeverywhere.com) are also available. Keys are stored in a file (called PGP keyring) and is passphrase protected. Key management is not centrally done, but done primarily on a peer-to-peer web-of-trust basis. PGP users can choose introducers whom them trust, and correspondingly accept the public keys of other users signed by these trusted introducers. We proceed to map the PGP's signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
PGP	M	L	L	L	M	L	M	M	M

4.4.2 PDF

The push towards secure electronic transactions saves costs for the organization, improves productivity for the business processes and reduces wastage for the environment. The use of PDF (portable document format) [134] as a standard for electronic document exchange is prevalent, and hundreds of countries have legislated the use of a digitally-signed PDF coupled with necessary audit and authentication processes to make the electronic document similarly recognizable as paper-based documents. Within ISO 32000 [134], it describes how the signature (either RSA up to 4096 bits, DSA up to 4096 bits) is to be PKCS#1 signed and PKCS#7 packed and embedded into the PDF document. Adobe has announced that their v11.x products [135] support ECDSA (NIST curves) but no PQ algorithms are planned in the pipeline.

ETSI (European Telecommunication Standards Institute) has released the PAdES (PDF Advanced Electronic Signature) standard [136] for organizations to comply with the eIDAS [4] regulation. Digital signature implementations that follow the ETSI standard will be deemed recognized as advanced electronic signatures (AES) under the eIDAS regulation. The ETSI standard defines multiple profiles for various use-cases such as Basic which is aligned with ISO 32000, Enhanced for inclusion of signature attributes and policy, and LTV (Long-term validation) for inclusion of the CRL (certificate revocation lists) and OCSP (online certificate status protocol) responses to extend the validity of the signature beyond the validity of the certificate.

Applying advanced electronic signatures (AES) on documents is meant to protect the integrity and authenticity of the document as required in many jurisdictions including USA, but non-repudiation is not a requirement. Typical AES implementations rely on the document server in the backend to digitally sign the document using a common key protected by a HSM after authenticating the signer. This is in contrast with qualified electronic signatures (QES) which requires the signer to have in possession the private key used for signing and provides for a higher level of "assurance". In QES implementations, signers will be issued with a USB token or security key to store the private key that is used for signing. We proceed to map the PDF signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
PDF-AES	L	L	M	L	L	M	M	M	L
PDF-QES	S	M	M	M	S	M	M	M	L

PDF-AES implementations allow for more resources for signing since it is server-based, while PDF-QES implementations require the signing to happen within the USB token in the possession of the signer.

4.4.3 Operating Systems

In order to ensure that only authorized applications and code are executed, operating systems support code-signing [137-139] as a means to allow developers and publishers to protect the integrity of the applications when deployed on the end-users' platforms. In addition, such applications can also make use of the hardware-based protection such as Intel SGX [140] and Trusted Execution Environment (TEE) on mobiles [141] for an added layer of execution attestation. Operating systems also rely on code-signing to ensure a secure boot process where only signed system modules are loaded and run.

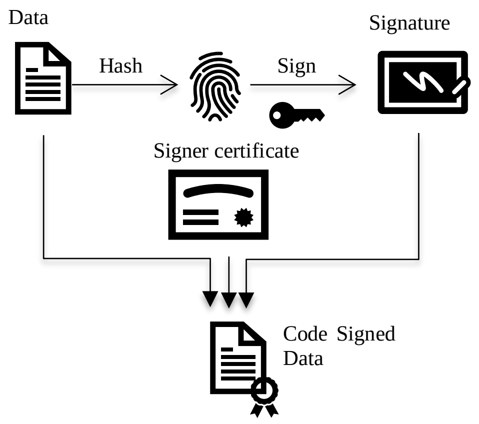

Figure 9. Code signing on iOS [138]

Figure 9 shows the simplified code-signing functionality on iOS, which is the operating system used in all Apple mobile and tablet devices. For a publisher to sign an app, it must first generate a code-signing key which is certified by the operating system. When an app is to be published, the publisher will hash the binary of the application and sign the hash using its private key. This app, with the signed hash and publisher certificate, can then be packaged together as a complete signed app which can be verified by the operating system when the end-user starts the app on the platform. The algorithms used for code signing are RSA and ECDSA.

We then proceed to map the code signing requirements into the signing parameters from Section 3.2.

	KeyGen Resource	Secret Key Size	Public Key Size	Key Lifetime	Sig Resource	Sig Size	Sig Time	Ver Resource	Sig lifetime
Code signing	L	L	L	M	L	L	L	M	L

We should take device considerations into account as IoT devices are also relying on signed firmware as part of their boot-up sequence, and hence signature verification is important for such use-cases.

5 Analysis

We are now ready to analyze the PQ algorithms from Section 3.2 against the applications listed in Section 4 using the signature parameters.

5.1 Scoring the Algorithms

The basis of scoring is as follows:

- For each parameter, if the algorithm provides an equal or closer to optimal than what the application allows for, then score remains unchanged.
- For each parameter, if the algorithm is one range worse (e.g. M instead of S) than what the application allows for, then subtract 1 from the score.

- If there is any parameter where the algorithm is two ranges worse (e.g. L instead of S) than what the application allows for, then deem the algorithm as unsuitable (NF = not fit), regardless of score.

A zero score means that there are no changes needed and the algorithm can most likely be used for the application. Following zero, the algorithm with the largest score (i.e. the least negative score) is the next most appropriate since it requires the least number of changes to be used by the application. Table 3 shows the scoring of each algorithm against the identified application.

Table 3. Scoring of Algorithm vs Application

	Dilithium	Falcon	qTESLA	GeMSS	LUOV	MQDSS	Rainbow	SPHINCS+	Picnic	XMSS /LMS H=20	XMSS /LMS H=10	Large RSA
3SKey	-3	-1	-4	NF	NF	NF	NF	NF	-2	NF	-1	NF
EMV (SDA)	-2	-1	NF	NF	NF	NF	NF	-2	NF	-2	NF	NF
EMV (DDA)	NF	-3	NF	NF	NF	NF	NF	NF	NF	NF	-3	NF
CA Key	0	0	-1	-1	-2	-1	-1	0	0	0	-1	-1
ICAO Doc Signer	-2	-1	NF	NF	NF	NF	NF	-1	NF	-1	-1	NF
GSM eSIM	NF	-1	NF	NF	NF	NF	NF	NF	-3	-1	0	NF
TLS server	0	0	0	NF	NF	NF	0	NF	-1	-2	NF	NF
TLS client	0	0	0	-3	-2	-2	-1	-4	0	-2	NF	-3
Bitcoin	0	0	0	-3	-3	-4	-1	-4	-1	-2	NF	-5
FIDO	0	0	0	-3	-3	-2	-1	-3	0	-2	-1	-3
PGP	0	0	0	-3	-3	-3	-1	-4	0	-2	NF	-3
PDF-AES	0	0	-1	-2	-3	-3	-1	-2	-1	-1	NF	-4
PDF-QES	-4	-1	-4	NF	NF	NF	NF	NF	-2	NF	-1	NF
Code signing	0	0	0	0	-1	-1	0	0	0	0	-1	0

5.2 Fitting the Algorithms to Applications

With the scoring, we next fit the algorithms on their suitability based on their overall ability to meet the applications requirements. The top 4 algorithms, by only counting a full fit (i.e. score 0) a partial fit (i.e. a negative score) or not fit (i.e. NA score), are in Table 4:

Table 4. Top 4 best-fit algorithms

Algorithm	Applications fully fit	Applications partially fit	Applications not fit

Falcon	CA Key, TLS Server, TLS Client, Bitcoin, FIDO, PGP, PDF (AES), Code Signing	3SKey, EMV (SDA), EMV (DDA), ICAO Doc Signer, GSM eSIM, PDF (QES)	
Dilithium	CA Key, TLS Server, TLS Client, Bitcoin, FIDO, PGP, PDF (AES), Code Signing	3SKey, EMV (SDA), ICAO Doc Signe, PDF (QES)	EMV (DDA), GSM eSIM
Picnic	CA Key, TLS Client, FIDO, PGP, Code Signing	3SKey, GSM eSIM,, TLS Server, Bitcoin, PDF (AES), PDF (QES)	EMV (SDA), EMV (DDA), IACO Doc Signer
XMSS/LMS H=20	CA Key, Code Signing	EMV (SDA), ICAO Doc Signer, GSM eSIM,, TLS Server, TLS Client, Bitcoin, FIDO, PGP, PDF (AES)	3SKey, EMV(DDA), PDF (QES)

If we next attempt to improve the 4 above algorithms with an appropriate complementary algorithm (e.g. XMSS/LMS H=10 or SPHINCS+), we see the following results in Table 5:

Table 5. Top 4 best-fit algorithms with supplementary algorithm

Algorithm	Applications fully fit	Applications partially fit	Applications not fit
Falcon + XMSS/LMS H=10	CA Key, GSM eSIM, TLS Server, TLS Client, Bitcoin, FIDO, PGP, PDF (AES), Code Signing	3SKey, EMV (SDA), EMV (DDA), ICAO Doc Signer, PDF (QES)	
Dilithium + XMSS/LMS H=10	CA Key, GSM eSIM, TLS Server, TLS Client, Bitcoin, FIDO, PGP, PDF (AES), Code Signing	3SKey, EMV (SDA), EMV (DDA), ICAO Doc Signer, PDF (QES)	
Picnic + XMSS/LMS H=10	CA Key, GSM eSIM, TLS Client, FIDO, PGP, Code Signing	3SKey, EMV (DDA), ICAO Doc Signer, TLS Server, Bitcoin, PDF (AES), PDF (QES)	EMV (SDA)
Picnic + SPHINCS+	CA Key, TLS Client, FIDO, PGP, Code Signing	3SKey, EMV (SDA), IACO Doc Signer, GSM eSIM,, TLS Server, Bitcoin, PDF (AES), PDF (QES)	EMV (DDA)
XMSS/LMS H=20 + XMSS/LMS H=10	CA Key, GSM eSIM, Code Signing	3SKey, EMV (SDA), EMV (DDA), ICAO Doc Signer, TLS Server, TLS Client, Bitcoin, FIDO, PGP, PDF (AES), PDF (QES)	

5.3 Call to Action

With the best-fit algorithms identified, we next scope the additional improvements needed to the algorithms to close the gap for a full fit. Table 6 shows the specific improvements needed on the algorithms to fit each of the 11 applications.

Table 6. Improvements needed for algorithms

	Falcon	Falcon + XMSS/LMS H=10	Dilithium + XMSS/LMS H=10	XMSS/LMS H=20 + XMSS/LMS H=10
3SKey	Falcon KeyGen → S	Falcon KeyGen → S Or XMSS/LMS H=10 Key lifetime → M	XMSS/LMS H=10 Key lifetime → M	XMSS/LMS H=10 Key lifetime → M
EMV (SDA)	Falcon Public keysize → S	Falcon Public keysize → S	Dilithium Public keysize → S, Sigsize → S	XMSS/LMS H=20 Key lifetime → L, Sigsize → S
EMV (DDA)	Falcon KeyGen →S, Secret keysize →S, Public keysize → S	Falcon KeyGen →S, Secret keysize →S, Public keysize → S Or	XMSS/LMS H=10 Key lifetime → M, Sigsize→S, Sigtime→S	XMSS/LMS H=10 Key lifetime → M, Sigsize→S, Sigtime→S

		XMSS/LMS H=10 Key lifetime → M, Sigsize→S, Sigtime→S		
CA Key	Fit	Fit	Fit	Fit
ICAO Doc Signer	Falcon Public keysize→S	Falcon Public keysize→S Or XMSS/LMS H=10 Sigsize→S	XMSS/LMS H=10 Sigsize→S	XMSS/LMS H=20 Sigsize→S or XMSS/LMS H=10 Sigsize→S
GSM eSIM	Falcon Secret keysize→S	Fit	Fit	Fit
TLS server	Fit	Fit	Fit	XMSS/LMS H=20 Key lifetime→L, Sigtime→S
TLS client	Fit	Fit	Fit	XMSS/LMS H=20 Keygen→M, Key lifetime→L
Bitcoin	Fit	Fit	Fit	XMSS/LMS H=20 Keygen→M, Key lifetime→L
FIDO	Fit	Fit	Fit	XMSS/LMS H=10 Key lifetime→M
PGP	Fit	Fit	Fit	XMSS/LMS H=20 Keygen→M, Key lifetime→L
PDF-AES	Fit	Fit	Fit	XMSS/LMS H=20 Key lifetime→L
PDF-QES	Falcon Keygen →S	Falcon Keygen →S or XMSS/LMS H=10 Key lifetime→M	XMSS/LMS H=10 Key lifetime→M	XMSS/LMS H=10 Key lifetime→M
Code signing	Fit	Fit	Fit	Fit

6 Conclusion

This study has done a review of the digital signature algorithms uncovered through the NIST PQC standardization contest with the community and attempted to map each algorithm against existing applications that require them. We used a parameterized model on specific performance and size characteristics to score the fit, with an underlying assumption that the signature algorithms should be sufficiently secure.

Based on the model, we are able to use the gaps identified between what is available versus what is needed to focus the research towards practical translation of these algorithms. These include:

- Focus on Falcon & XMSS/LMS H=10 as the best-fit combination amongst all the algorithms.
- Optimization of Falcon to reduce the size of keys and the resources to generate the keys.
- Optimization of XMSS/LMS H=10, H=20 to reduce the signature size, time taken to sign and increase the number of signatures per key.

REFERENCES

[1] ITU. (2016). *X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*. Available: https://www.itu.int/rec/T-REC-X.509-201610-I/en

[2] E. Rescorla. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. Available: https://tools.ietf.org/html/rfc8446

[3] EMVco. (2011). *EMV Integrated Circuit Card Specifications for Payment Systems. Book 2 - Security and Key Management v4.3*. Available: https://www.emvco.com/wp-content/uploads/2017/05/EMV_v4.3_Book_2_Security_and_Key_Management_20120607061923900.pdf

[4] EuropeanUnion. (2014). *REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 July 2014*. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG

[5] U. S. Government. (2000). *Pub. L. 106-229 Electronic Signatures in Global and National Commerce Act*. Available: https://www.govinfo.gov/content/pkg/PLAW-106publ229/pdf/PLAW-106publ229.pdf

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[7] K. Neal, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, 1987.

[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303-332, 1999.

[9] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *arXiv preprint quant-ph/0301141*, 2003.

[10] J. Aron. (2019). *IBM unveils its first commercial quantum computer*. Available: https://www.newscientist.com/article/2189909-ibm-unveils-its-first-commercial-quantum-computer/

[11] F. Lardinois. (2019). *IBM will soon launch a 53-qubit quantum computer*. Available: https://techcrunch.com/2019/09/18/ibm-will-soon-launch-a-53-qubit-quantum-computer/

[12] L. Chen *et al.*, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[13] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical review letters*, vol. 79, no. 2, p. 325, 1997.

[14] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM journal on Computing*, vol. 26, no. 5, pp. 1510-1523, 1997.

[15] D. Moody. (2017). *The Ship has Sailed: The NIST Post-Quantum Cryptography "Competition"*. Available: https://csrc.nist.gov/Presentations/2017/The-Ship-has-Sailed-The-NIST-Post-Quantum-Cryptog

[16] C. F. Kerry and C. R. Director, "Fips pub 186-4 federal information processing standards publication digital signature standard (DSS)," 2013.

[17] T. U. Eindhoven. (2015). *Initial recommendations of long-term secure post-quantum systems*. Available: http://pqcrypto.eu.org/docs/initial-recommendations.pdf

[18] P. Wallden and E. Kashefi, "Cyber security in the quantum era," *Commun. ACM*, vol. 62, no. 4, p. 120, 2019.

[19] NIST. (2016). *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. Available: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

[20] NIST. (2019). *Post-Quantum Cryptography*. Available: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions

[21] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Annual International Cryptology Conference*, 1996, pp. 104-113: Springer.

[22] C. G. Günther, "An identity-based key-exchange protocol," in *Workshop on the Theory and Application of of Cryptographic Techniques*, 1989, pp. 29-37: Springer.

[23] T. Dierks and E. Rescorla. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2*. Available: https://tools.ietf.org/html/rfc5246

[24] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99-108: ACM.

[25] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Annual International Cryptology Conference*, 1997, pp. 112-131: Springer.

[26] P. Nguyen, "Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from crypto'97," in *Annual International Cryptology Conference*, 1999, pp. 288-304: Springer.

[27] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*, 1998, pp. 267-288: Springer.

[28] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[29] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," *J. ACM*, vol. 60, no. 6, pp. 1-35, 2013.

[30] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 197-206: ACM.

[31] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSIGN: Digital signatures using the NTRU lattice," in *Cryptographers' Track at the RSA Conference*, 2003, pp. 122-140: Springer.

[32] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," presented at the Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques, French Riviera, France, 2010.

[33] V. Lyubashevsky, "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures," presented at the Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, Tokyo, Japan, 2009.

[34] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," presented at the Proceedings on Advances in cryptology---CRYPTO '86, Santa Barbara, California, USA, 1987.
[35] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. (2018). *Crystals–dilithium: Digital signatures from Module Lattices*. Available: https://cryptojedi.org/papers/dilithium-20170627.pdf
[36] S. Bai and S. D. Galbraith, "An improved compression technique for signatures based on learning with errors," in *Cryptographers' Track at the RSA Conference*, 2014, pp. 28-47: Springer.
[37] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2012, pp. 530-547: Springer.
[38] L. Ducas and T. Prest, "Fast fourier orthogonalization," in *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, 2016, pp. 191-198: ACM.
[39] P.-A. Fouque *et al.* (2018). *Falcon: Fast-Fourier lattice-based compact signatures over NTRU*. Available: https://falcon-sign.info/falcon.pdf
[40] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114-116, 1978.
[41] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384-386, 1978.
[42] D. J. Bernstein, "Grover vs. mceliece," in *International Workshop on Post-Quantum Cryptography*, 2010, pp. 73-80: Springer.
[43] M. R. Garey and D. S. Johnson, *Computers and intractability*. wh freeman New York, 2002.
[44] N. Courtois, L. Goubin, W. Meier, and J.-D. Tacier, "Solving underdefined systems of multivariate quadratic equations," in *International Workshop on Public Key Cryptography*, 2002, pp. 211-227: Springer.
[45] T. Matsumoto and H. Imai, "Public quadratic polynomial-tuples for efficient signature-verification and message-encryption," in *Workshop on the Theory and Application of of Cryptographic Techniques*, 1988, pp. 419-453: Springer.
[46] J. Patarin, "Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88," in *Annual International Cryptology Conference*, 1995, pp. 248-261: Springer.
[47] J. Patarin, "Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 1996, pp. 33-48: Springer.
[48] J. Patarin, "The oil and vinegar signature scheme," in *Dagstuhl Workshop on Cryptography September, 1997*, 1997.
[49] A. Kipnis and A. Shamir, "Cryptanalysis of the HFE public key cryptosystem by relinearization," in *Annual International Cryptology Conference*, 1999, pp. 19-30: Springer.
[50] A. Kipnis and A. Shamir, "Cryptanalysis of the oil and vinegar signature scheme," in *Annual International Cryptology Conference*, 1998, pp. 257-266: Springer.
[51] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced oil and vinegar signature schemes," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 1999, pp. 206-222: Springer.
[52] M.-L. Akkar, N. T. Courtois, R. Duteuil, and L. Goubin, "A fast and secure implementation of Sflash," in *International Workshop on Public Key Cryptography*, 2003, pp. 267-278: Springer.
[53] B. Preneel *et al.*, "Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity, and Encryption," *Berlin Heidelberg NewYork London Paris Tokyo Hong Kong Barcelona Budapest: Springer-Verlag*, 2004.
[54] V. Dubois, P.-A. Fouque, A. Shamir, and J. Stern, "Practical cryptanalysis of SFLASH," in *Annual International Cryptology Conference*, 2007, pp. 1-12: Springer.
[55] G. Alagic *et al.*, *Status report on the first round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, National Institute of Standards and Technology, 2019.
[56] J. Patarin, N. Courtois, and L. Goubin, "Quartz, 128-bit long digital signatures," in *Cryptographers' Track at the RSA Conference*, 2001, pp. 282-297: Springer.
[57] A. Casanova, J.-C. Faug'ere, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem, "GeMSS: A Great Multivariate Short Signature," PhD thesis, UPMC-Paris 6 Sorbonne Universités, 2017.
[58] W. Beullens, A. Szepieniec, F. Vercauteren, and B. Preneel, "LUOV: Signature scheme proposal for NIST PQC project," 2017.
[59] K. Sakumoto, T. Shirai, and H. Hiwatari, "Public-key identification schemes based on multivariate quadratic polynomials," in *Annual Cryptology Conference*, 2011, pp. 706-723: Springer.
[60] S. M. E. Y. Alaoui, Ö. Dagdelen, P. Véron, D. Galindo, and P.-L. Cayrel, "Extended security arguments for signature schemes," in *International Conference on Cryptology in Africa*, 2012, pp. 19-34: Springer.
[61] A. Hülsing, J. Rijneveld, S. Samardjiska, and P. Schwabe, "From 5-pass MQ-based identification to MQ-based signatures," *IACR Cryptology ePrint Archive*, vol. 2016, p. 708, 2016.
[62] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *International Conference on Applied Cryptography and Network Security*, 2005, pp. 164-175: Springer.
[63] J. Ding, M. Chen, A. Petzoldt, D. Schmidt, and B. Yang, "Rainbow-Algorithm Specification and Documentation," ed, 2017.
[64] L. Lamport, "Constructing digital signatures from a one-way function," Technical Report CSL-98, SRI International Palo Alto1979.
[65] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *International workshop on fast software encryption*, 2004, pp. 371-388: Springer.
[66] A. Hülsing, "W-OTS+–shorter signatures for hash-based signature schemes," in *International Conference on Cryptology in Africa*, 2013, pp. 173-188: Springer.
[67] R. C. Merkle, "A certified digital signature," in *Conference on the Theory and Application of Cryptology*, 1989, pp. 218-238: Springer.
[68] A. Huelsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen. (2018). *XMSS: eXtended Merkle Signature Scheme*. Available: https://tools.ietf.org/html/rfc8391
[69] D. McGrew, M. Curcio, and S. Fluhrer. (2019). *Leighton-Micali Hash-Based Signatures*. Available: https://tools.ietf.org/html/rfc8554

[70] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Australasian Conference on Information Security and Privacy*, 2002, pp. 144-153: Springer.
[71] D. J. Bernstein *et al.*, "SPHINCS: practical stateless hash-based signatures," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015, pp. 368-397: Springer.
[72] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
[73] D. Bernstein *et al.*, "SPHINCS+–Submission to the NIST post-quantum project," ed: Dec, 2017.
[74] A. Rostovtsev and A. Stolbunov, "Public-Key Cryptosystem Based on Isogenies," *IACR Cryptology ePrint Archive,* vol. 2006, p. 145, 2006.
[75] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory,* vol. 22, no. 6, pp. 644-654, 1976.
[76] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," in *Proceedings of STOC 1985*, 1985, pp. 291-304.
[77] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology,* journal article vol. 1, no. 2, pp. 77-94, June 01 1988.
[78] J.-J. Quisquater *et al.*, "How to explain zero-knowledge protocols to your children," in *Conference on the Theory and Application of Cryptology*, 1989, pp. 628-631: Springer.
[79] J. Camenisch and R. Chaabouni, "Efficient protocols for set membership and range proofs," in *2008 International Conference on the Theory and Application of Cryptology and Information Security*, 2008, pp. 234-252: Springer.
[80] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy*, 2018, pp. 315-334: IEEE.
[81] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," *Journal of the ACM (JACM),* vol. 38, no. 3, pp. 690-728, 1991.
[82] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 21-30: ACM.
[83] I. Giacomelli, J. Madsen, and C. Orlandi, "Zkboo: Faster zero-knowledge for boolean circuits," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 1069-1083.
[84] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," presented at the Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, Cambridge, Massachusetts, 2012.
[85] M. Chase *et al.*, "Post-quantum zero-knowledge and signatures from symmetric-key primitives," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1825-1842: ACM.
[86] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner, "Ciphers for MPC and FHE," Berlin, Heidelberg, 2015, pp. 430-454: Springer Berlin Heidelberg.
[87] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.
[88] D. Unruh, "Non-interactive zero-knowledge proofs in the quantum random oracle model," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015, pp. 755-784: Springer.
[89] M. Chase *et al. The picnic signature algorithm specification (2019)*. Available: https://github.com/microsoft/Picnic/blob/master/spec/spec-v2.1.pdf
[90] NIST. (2018). *Stateful Hash-Based Signatures*. Available: https://csrc.nist.gov/Projects/Stateful-Hash-Based-Signatures
[91] F. T. Leighton and S. Micali, "Large provably fast and secure digital signature schemes based on secure hash functions," ed: Google Patents, 1995.
[92] NIST, "Stateful Hash-Based Signatures: Public Comments on Misuse Resistance—Request Issued Feb. 4, 2019," 2019.
[93] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS-a practical forward secure signature scheme based on minimal security assumptions," in *International Workshop on Post-Quantum Cryptography*, 2011, pp. 117-129: Springer.
[94] R. Cleve, "The query complexity of order-finding," in *Proceedings 15th Annual IEEE Conference on Computational Complexity*, 2000, pp. 54-59: IEEE.
[95] S. Beauregard, "Circuit for Shor's algorithm using 2n+ 3 qubits," *arXiv preprint quant-ph/0205095,* 2002.
[96] Y. Takahashi and N. Kunihiro, "A quantum circuit for shor's factoring algorithm using 2n + 2 qubits," *Quantum Info. Comput.,* vol. 6, no. 2, pp. 184-192, 2006.
[97] D. J. Bernstein, N. Heninger, P. Lou, and L. Valenta, "Post-quantum RSA," in *International Workshop on Post-Quantum Cryptography*, 2017, pp. 311-329: Springer.
[98] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal blind quantum computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009, pp. 517-526: IEEE.
[99] C. H. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 1984, vol. 175, p. 8.
[100] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Physical review letters,* vol. 67, no. 6, p. 661, 1991.
[101] D. Gottesman and I. Chuang, "Quantum digital signatures," *arXiv preprint quant-ph/0105032,* 2001.
[102] M.-Q. Wang, X. Wang, and T. Zhan, "An efficient quantum digital signature for classical messages," *Quantum Information Processing,* vol. 17, no. 10, p. 275, 2018.
[103] P. J. Clarke, R. J. Collins, V. Dunjko, E. Andersson, J. Jeffers, and G. S. Buller, "Experimental demonstration of quantum digital signatures using phase-encoded coherent states of light," *Nature communications,* vol. 3, p. 1174, 2012.
[104] W. Li, R. Shi, and Y. Guo, "Blind quantum signature with blind quantum computation," *International Journal of Theoretical Physics*, vol. 56, no. 4, pp. 1108-1115, 2017.
[105] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing,* vol. 17, no. 2, pp. 281-308, 1988.
[106] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," Cham, 2014, pp. 313-326: Springer International Publishing.
[107] M. Bellare and P. Rogaway, "The exact security of digital signatures-How to sign with RSA and Rabin," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 1996, pp. 399-416: Springer.

[108] T. Jager, S. A. Kakvi, and A. May, "On the security of the PKCS# 1 v1. 5 signature scheme," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1195-1208: ACM.
[109] "ROCA Vulnerability and eID: Lessons Learned," Available: https://www.ria.ee/sites/default/files/content-editors/kuberturve/roca-vulnerability-and-eid-lessons-learned.pdf
[110] SWIFT. *3SKey: The multibank, multichannel personal digital identity solution*. Available: https://www.swift.com/our-solutions/corporates/control/order-3skey
[111] SWIFT. (2016). *How much do you pay for your PKI solution?* Available: https://www.swift.com/our-solutions/corporates/control/order-3skey/document-centre?tl=en#topic-tabs-menu
[112] EMVco. (2018). *EMVCo Reports Over Half of Cards Issued Globally are EMV®-enabled* Available: https://www.emvco.com/wp-content/uploads/2018/04/Global-Circulation-Figures_FINAL.pdf
[113] ISO. (2013). *ISO/IEC 7816-4:2013 Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*. Available: https://www.iso.org/standard/54550.html
[114] D. Van Geest and S. Fluhrer. (2018). *Algorithm Identifiers for HSS and XMSS for Use in the Internet X.509 Public Key Infrastructure*. Available: https://tools.ietf.org/id/draft-vangeest-x509-hash-sigs-01.html
[115] VASCO. (2011). *VASCO Announces Bankruptcy Filing by DigiNotar B.V.* Available: https://web.archive.org/web/20110923180445/http://www.vasco.com/company/press_room/news_archive/2011/news_vasco_announces_bankruptcy_filing_by_diginotar_bv.aspx
[116] ICAO. (2015). *Doc 9303: Machine Readable Travel Documents*. Available: https://www.icao.int/publications/pages/publication.aspx?docnum=9303
[117] GSMA. (2018). *eSIM Whitepaper: The what and how of Remote SIM Provisioning*. Available: https://www.gsma.com/esim/wp-content/uploads/2018/06/eSIM-Whitepaper-v4.11.pdf
[118] GSMA. (2018). *RSP Technical Specification Version 2.2.1*. Available: https://www.gsma.com/newsroom/wp-content/uploads//SGP.22-v2.2.1-2.pdf
[119] I. K. Center. (2019). *An overview of the SSL or TLS handshake*. Available: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm
[120] Cloudflare. *What happens in a TLS Handshake ?* Available: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/
[121] M. D. Green and I. Miers, "Forward secure asynchronous messaging from puncturable encryption," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 305-320: IEEE.
[122] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Available: https://bitcoin.org/bitcoin.pdf
[123] A. De Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801-805, 2018.
[124] M. Qu, "SEC 2: Recommended elliptic curve domain parameters," *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6*, 1999.
[125] P. Wuille. (2017). *BIP 0032 - Hierarchical Deterministic Wallets*. Available: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki
[126] M. Palatinus and P. Rusnak, "BIP 0044 - Multi-Account Hierarchy for Deterministic Wallets," Available: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki
[127] K. Chalkias, J. Brown, M. Hearn, T. Lillehagen, I. Nitto, and T. Schroeter, "Blockchained post-quantum signatures," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1196-1203: IEEE.
[128] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, 2017, pp. 357-388: Springer.
[129] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv preprint arXiv:1805.04548,* 2018.
[130] S. Machani, R. Philpott, S. Srinivas, J. Kemp, and J. Hodges. (2017). *FIDO UAF Architectural Overview*. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-ps-20170202/fido-uaf-overview-v1.1-ps-20170202.pdf
[131] P. Zimmermann, "Why i wrote pgp," *Part of the Original*, 1991.
[132] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. (2007). *OpenPGP Message Format*. Available: https://tools.ietf.org/html/rfc4880
[133] A. Jivsov. (2012). *Elliptic Curve Cryptography (ECC) in OpenPGP*. Available: https://tools.ietf.org/html/rfc6637
[134] ISO. (2013). *ISO 32000-1:2008 Document management — Portable document format — Part 1: PDF 1.7*. Available: https://www.iso.org/standard/51502.html
[135] Adobe. (2018). *Adobe DC Digital Signatures Guide >> Supported Standards*. Available: https://www.adobe.com/devnet-docs/acrobatetk/tools/DigSigDC/standards.html
[136] ETSI. (2009). *ETSI TS 102 778-1 V1.1.1 Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - a framework document for PAdES*. Available: https://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf
[137] Microsoft. (2017). *Introduction to Code Signing*. Available: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms537361(v=vs.85)
[138] Apple. *Code Signing Guide*. Available: https://developer.apple.com/library/archive/documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html
[139] Android. *Application Signing*. Available: https://source.android.com/security/apksigning
[140] Intel. (2014). *Intel Software Guard Extensions Programming Reference*. Available: https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf
[141] Trustonic. *Trustonic Device Security*. Available: https://www.trustonic.com/solutions/trustonic-secured-platforms-tsp/