Layering Quantum-Resistance into Classical Digital Signature Algorithms

Teik Guan Tan(✉) and Jianying Zhou

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

Abstract. It is proven that asymmetric key cryptographic systems that rely on Integer Factorization or Discrete Logarithm as the underlying hard problem are vulnerable to quantum computers. Using Shor's algorithm on a large-enough quantum computer, an attacker can cryptanalyze the public key to obtain the private key in $O(logN)$ time complexity. For systems that use the classical Digital Signature Algorithm (DSA), Rivest-Shamir-Adleman (RSA) algorithm or Elliptic-Curve Digital Signature Algorithm (ECDSA), it means that authentication, data integrity and non-repudiation between the communicating parties cannot be assured in the post-quantum era.

In this paper, we present a novel approach using zero-knowledge proofs on the pre-image of the private signing key to layer in quantum-resistance into digital signature deployments that require longer-term post-quantum protection while maintaining backward compatibility with existing implementations. We show that this approach can extend the cryptographic protection of data beyond the post-quantum era and is also easy to migrate to. An implementation of this approach applying a ZKBoo zero-knowledge proof on ECDSA signatures is realized using a RFC3161-compatible time-stamp server with OpenSSL and an Adobe Acrobat Reader DC.

Keywords: Digital signature · Elliptic Curve Digital Signature Algorithm (ECDSA) · Zero-knowledge proof · Post-quantum security

1 Introduction

Asymmetric key cryptography is the tool used by systems worldwide to preserve trust amongst parties in the digital realm. The use of digital signatures allow communicating parties to authenticate each other, check the integrity of the data exchanged, and prove the origin of the data in situations of repudiation. Under National Institute of Standards and Technology's (NIST) Digital Signature Standards FIPS 186-4 [26], three signature algorithms are described. These are i) Digital Signature Algorithm (DSA) which is based on discrete logarithm cryptography first introduced by Diffie and Hellman [15]; ii) Rivest-Shamir Adelman (RSA) [34], and iii) Elliptic-Curve Digital Signature Algorithm (ECDSA)

J. K. Liu et al. (Eds.): ISC 2021, LNCS 13118, pp. 26–41, 2021.
https://doi.org/10.1007/978-3-030-91356-4_2

which is based on Elliptic Curve Cryptography (ECC) [8] and together we call them classical digital signature algorithms. The security of DSA and ECDSA are based on the hard problem of solving discrete logarithm over a finite field of very large numbers, while the security of RSA is based on the difficulty of integer factorization over a finite field of very large numbers.

The advent of large fault-tolerant quantum computers poses a big risk to systems that use these digital signature algorithms. Shor's [35] algorithm is able to solve both the discrete logarithm problem and integer factorization problem in $O(logN)$ polynomial time. This means that any adversary in possession of a large-enough quantum computer will be able to compute a user's private signing key when given the user's public key in a matter of hours, and generate valid digital signatures to impersonate the user. In addition, data that was previously signed by the user no longer can be proven to be authentic and trustworthy. As a reference post-quantum deadline, NIST has provided a report [13] mentioning that by year 2030, it is likely that a quantum computer capable of cryptanalyzing RSA-2048 can be built with a budget of one billion dollars. To address this, NIST is embarking on a post-quantum standardization exercise [29,30] to select suitable quantum-secure digital signature and key-exchange algorithms. The final selection is expected to complete soon with the new standards slated to be published by year 2024. Separately, NIST has also recommended two stateful hash-based signatures, namely Leighton-Micali Signatures and eXtended-Merkle Signature Scheme, for post-quantum use under conditions [14].

While the industry is likely to encourage new system implementations post-2024 to consider adopting the new digital signature standards, we expect different challenges for existing or upcoming systems. NIST has published some challenges they explored with post-quantum cryptography replacement and migration [4,12], and we supplement it with additional questions specific to digital signatures. Should system operators using digital signatures embark on a cryptographic migration to the stateful hash-based signatures [14] instead of waiting for the post-quantum standardization? How about documents that are already digitally signed, and are required to remain trustworthy beyond year 2030? Do these documents need to be counter-signed with new algorithms? Since the counter-signer may be a non-interested third party to the transaction, what liability does the counter-signer bear for the verifying party? How about legacy systems that cannot be migrated? When are the verifying parties expected to be ready to verify the new algorithms since the migrations are happening at a different pace? What are the legal implications for the verifying party if the existing non-quantum-secure signature passes verification, but the verifying party is unable to verify the new quantum-secure signature? These are questions with no straightforward answers and seeking a proper resolution may require more time than afforded by the impending post-quantum deadline.

Our approach is different. If the existing digital signatures can remain quantum-resistant even after large-enough quantum computers are built, then many of the transition-related questions can be avoided. Existing systems will not face compatibility issues, migration timelines to the new algorithms are less

counter-party dependent, and existing digitally signed documents retain their authenticity in the post-quantum era. This is possible by layering a quantum-secure zero-knowledge proof of the pre-image of the private signing key along with the signature. Our contributions are as follows:

- Extend the digital signature scheme to construct a quantum-resistant digital signature scheme with backward-compatibility properties.
- Realize the quantum-resistant digital signature scheme using a zero-knowledge proof to be included with digital signatures to make them quantum-resistant.
- Deploy a real-world implementation of a RFC3161-compatible [2] time-stamp server to issue quantum-resistant ECDSA timestamp digital signatures with X.509v3 certificates that are compatible with the existing Adobe PDF Acrobat Reader DC v2021.x.

The rest of this paper is organized as follows. Section 2 covers the background of digital signatures and zero-knowledge proofs. Section 3 describes the proposed signature scheme, covers the description of the algorithms and provides measurements made on execution timings and proof sizes. Section 4 describes the real-line deployment of the proposed signature scheme and covers the migration strategy. Section 5 discusses some of the related works and Sect. 6 concludes the paper.

2 Background

2.1 Digital Signature Basics

We describe a simple scenario for two communicating parties Alice and Bob, where Alice has a message M to be sent to Bob. Alice wants to ensure that Bob receives the message unchanged (integrity) and knows that it is from Alice (authenticity). Bob wants to be able to prove to a third-party that the message is indeed from Alice (non-repudiation).

Definition 1. *We define a digital signature scheme as a triple of polynomial-time algorithms KeyGen, Sign, Verify with the following parameters:*

$KeyGen(1^n) \Rightarrow \{K_s, K_p\}$ *takes in a security parameter* 1^n *which defines the cryptographic key strength of* n*, and outputs a private key* K_s *and corresponding public key* K_p*.*

$Sign(M, K_s) \Rightarrow \{\sigma\}$ *takes in a message* M *and the private key* K_s*, and outputs a signature* σ*.*

$Verify(M, K_p, \sigma) \Rightarrow \{result\}$ *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* $Sign(M, K_s)$*.*

In this case, Alice, the signing party, calls $KeyGen$ to generate $\{K_s, K_p\}$. K_p is published where Bob and other parties have access to. Alice then calls $Sign$ with her private key K_s to sign the message M, generating a signature

σ. Alice transmits $\{M, \sigma\}$ to Bob. Bob, the verifying party, calls $Verify$ with Alice's public key K_p to verify the signature σ for message M. If $Verify$ returns accept, then Bob has successfully received a message M unchanged and the signature proof σ from Alice.

2.2 Zero-Knowledge Proof

Goldwasser et al. [20] provided the concept of zero-knowledge proofs where the proof conveys no additional knowledge besides the correctness of the proposition. While there has been many concrete realizations of zero-knowledge proofs, quantum-resistant non-interactive zero-knowledge proofs are either ZKStark [6] or MPC-in-the-head (Multi-party computation in-the-head) [24] based proofs.

For MPC-in-the-head proofs, a prover has to create a boolean computational circuit of n branches with commitment, of which $n-1$ views can be revealed to the verifier as proof of knowledge. To make the proof non-interactive, the prover can use Fiat-Shamir's heuristic [17] to deterministically, yet unpredictably decide which $n-1$ views to send to the verifier. The verifier then walks through the $n-1$ views with a $\frac{1}{n}$ chance that the proposition is incorrect. By increasing the number of rounds (with different random input parameters) that the prover has to compute the circuit and provide the views, it exponentially reduces the statistical probability that the prover is making a false claim.

3 Proposed Quantum-Resistant Digital Signatures

Since Shor's algorithm on quantum computers break the integer factorization problem [35], discrete logarithm problem [35] and elliptic-curve discrete logarithm problem [32], we can safely assume that adversaries can feasibly compute all RSA/DSA/ECDSA private keys K_s given the public key K_p when large enough quantum computers are built. On the other hand, symmetric key and hash-based cryptography remain relatively quantum-resistant. Grover's algorithm [21] on quantum computers can only achieve a quadratic speedup of $O(\sqrt{N})$ when performing a brute-force search, and this has been proven to be optimal [7].

Therefore, our proposal is to extend the signing process to layer in a zero-knowledge proof of knowledge of the pre-image of the private key to protect the signature. The extended verifying process can then verify this proof to ascertain that the signature is genuinely created by the owner of the private key and not a quantum-capable adversary. For backward-compatibility, the existing verifying process can still verify the digital signature without the proof, albeit losing the quantum-resistant assurance.

3.1 Quantum-Resistant Digital Signature Scheme

We start by extending the classical digital signature scheme (Definition 1) described in Sect. 2.1.

Definition 2. *The extended quantum-resistant digital signature scheme is as follows:*

$KeyGen_q(1^n) \Rightarrow \{\rho, K_p\}$ *takes in a security parameter* 1^n *which defines the cryptographic key strength of* n*, and outputs a secret pre-image* ρ *and a public key* K_p*.* K_p *is the associated public key to the private key* $H(\rho)$ *where* $H()$ *is a collapsing hash function [38].*

$Sign_q(M, \rho) \Rightarrow \{\sigma, \pi\}$ *takes in a message* M *and the secret pre-image* ρ*, and outputs a signature* σ *computed using* $Sign(M, H(\rho))$ *as well as a quantum-resistant zero-knowledge proof* π *that i)* $H(\rho)$ *is computed from* ρ *and ii)* σ *is computed from* $H(\rho)$*.*

$Verify_q(M, K_p, \sigma, \pi) \Rightarrow \{result\}$ *takes in a message* M*, the public key* K_p *and signature* σ*, and outputs accept if and only if* $Verify(M, K_p)$ *returns accept and* π *is a valid zero-knowledge proof that* σ *is computed from* ρ*.*

Intuitively, Definition 2 inherits the classical security properties of Definition 1 with an additional layer of quantum-resistance placed on the private key. A classical adversary will not be able to compromise the soundness of $Verify_q$ when interacting with the signing party since the additional information obtained from $Sign_q$ is a zero-knowledge proof that does not reveal the secret pre-image ρ or private key $K_s = H(\rho)$.

Lemma 1 (Quantum Resistance). *Definition 2 offers additional quantum-resistance for digital signatures generated using* $Sign_q$ *provided* $Verify_q$ *is used to verify the signature* σ *and proof* π*.*

Proof. We assume that a quantum-capable adversary is able to use Shor's algorithm [35] to recover $H(\rho)$ from K_p. Using $H(\rho)$, the adversary is then able to arbitrarily generate valid signatures σ using $Sign$ which will be accepted by $Verify$. However, the adversary will not be able generate the proof π since the value of ρ is not recoverable from $H(\rho)$ as $H()$ is a collapsing hash function and resistant to pre-image attacks even from quantum computers [38]. Thus, $Verify_q$ is resistant to quantum-capable adversaries. □

Lemma 2 (Backward Compatibility). *A signing party using* $KeyGen_q$ *and* $Sign_q$ *of Definition 2 generates signatures* σ *that are backward compatible with verifying parties using* $Verify$ *of Definition 1.*

Proof. Signatures σ returned by $Sign_q$ are generated using the same algorithm $Sign$ where $H(\rho)$ is effectively equal to K_s. Hence, any verifying party in this case using $Verify$ will be able to ignore π, and continue to call $Verify$ to check the validity of the signature σ with respect to M and K_p. A demonstration of the backward compatibility can be seen in Sect. 4. □

3.2 Realizing the Proposed Digital Signature Scheme

We use the following algorithms to realize our quantum-resistant digital signature scheme:

- *Digital signing algorithm.* Either DSA or ECDSA can be easily used as the digital signing algorithm. This is because the private key generator for DSA and ECDSA is essentially an unpredictable random number generated over a finite field. This matches nicely with the output of a one-way hash function $H()$. Using RSA as the signing algorithm is more complex and tedious since key generation involves the matching the output of a hash function to two or more unpredictable prime numbers used to compute the RSA modulus. Possible techniques include mapping the hash output into an ordered list of very large primes [25] or repeatedly hashing (or mining) random numbers till a prime is found. For our reference implementation, ECDSA is used as it has the smallest key size which translates to the smallest proof size. The curve chosen is secp256r1 (or prime256v1) [8].
- *Hash function.* The hash function to be used in our reference implementation is SHA-256 [18] as it is collapsing [38] and the output fits well with the secp256r1 curve.
- *Zero-knowledge proof system.* The zero-knowledge proof system to be used has to be post-quantum secure. We have chosen ZKBoo [19] as it is a 3-branch MPC-in-the-head realization and already has a ready SHA-256 implementation. ZKBoo is also used as the underlying proof system to create ZKB++ for Picnic [10], an alternative finalist candidate in NIST's post-quantum standardization exercise [30].

Realization of $KeyGen_q$. The function $KeyGen_q$ shown in Algorithm 1 works very similarly to $KeyGen$. An additional step (see Step 5 of Algorithm 1) is performed to hash the secret pre-image ρ prior to computing public key K_p.

Algorithm 1: Quantum-resistant ECDSA Key Generation $KeyGen_q$.

```
1 begin
2 |  G ← ECC base point; P ← ECC order;
3 |  Generate secret pre-image ρ;
4 |  Compute private key K_s = H(ρ);
5 |  Compute public key K_p = (x, y) where K_p ≡ K_s · G mod P;
6 |  destroy K_s;
7 |  return ρ, K_p;
8 end
```

Algorithm 2: Quantum-resistant ECDSA signing $Sign_q$.

begin
  $G \leftarrow$ ECC base point; $P \leftarrow$ ECC order;
  $\rho \leftarrow$ secret pre-image;
  $M \leftarrow$ message;
  Generate signature random r;
  Compute r^{-1} where $r * r^{-1} \equiv 1 \ mod \ P$;
  Compute $R = (R_x, R_y)$ where $R \equiv r \cdot G \ mod \ P$;
  Compute hash of message $H(M)$;
  Enumerate ZKBoo proof $\pi =$ **begin**
    Zero-knowledge computation of private key K_s where $K_s = H(\rho)$;
    Zero-knowledge computation of public key K_p where $K_p \equiv K_s \cdot G \ mod \ P$;
    Compute s where $s \equiv r^{-1} * (H(M) + R_x * K_s) \ mod \ P$;
    Commit R_x, s in the proof;
  end
  destroy r, r^{-1}, K_s;
  return $\sigma = \{R_x, s\}, \pi$;
end

Algorithm 3: Quantum-resistant ECDSA verification $Verify_q$.

begin
  $G \leftarrow$ ECC base point; $P \leftarrow$ ECC order;
  $K_p \leftarrow$ ECC public key;
  $R_x, s \leftarrow$ signature σ; $\pi \leftarrow$ proof; $M \leftarrow$ message;
  Compute s^{-1} where $s * s^{-1} \equiv 1 \ mod \ P$;
  Compute hash of message $H(M)$;
  Compute $u_1 = s^{-1} * H(M) \ mod \ P$;
  Compute $u_2 = s^{-1} * R_x \ mod \ P$;
  Compute $V = (V_x, V_y)$ where $V = u_1 \cdot G + u_2 \cdot K_p \ mod \ P$;
  if $V_x \neq R_x$ **then**
    return "Failed Signature Verification"
  end
  else
    Verify ZKBoo proof $\pi =$ **begin**
      Check that R_x, s is committed in the proof;
      Check that zero-knowledge computation of K_p from unknown pre-image is correct;
      if *Check Failed* **then**
        return "Failed Proof Verification"
      end
    end
  end
  return success;
end

Realization of $Sign_q$. The $Sign_q$ function is shown in Algorithm 2. Besides computing the ECDSA signature using the private key $H(\rho)$, the $Sign_q$ function

returns the ZKBoo proof π which includes: i) zero-knowledge proof of knowledge of pre-image of $H(\rho)$; ii) zero-knowledge proof that public key K_p is computed from $H(\rho)$; and iii) commitment that $H(M)$ is the message being signed.

The implementation in Step 10 of Algorithm 2 uses Giacomelli et al.'s [19] SHA-256 code. Special care has to be taken to code Step 11 of Algorithm 2 as the number of computational steps in the proof π could reveal the private key K_s. When performing elliptic-curve multiplication, we use the double-and-add always technique which is effective against side channel power analysis timing attacks [27].

Realization of $Verify_q$. The function $Verify_q$ shown in Algorithm 3 consists of two parts where the first part (from Steps 5 to 12) is the ECDSA signature verification similar to $Verify$ while the second part (from Steps 14 to 20) is the additional verification of the quantum-resistant zero-knowledge proof.

3.3 Performance Measurement

The proposed digital signature scheme is implemented in C[1] and tested on an Intel I5-8250U 8th Gen machine with 8 CPU cores and 8 GB RAM, running a Cygwin terminal on 64-bit Microsoft Windows 10. No operating system level CPU scheduling or adjustments are done. We measure the execution times of $Sign_q$ and $Verify_q$ as well as the proof sizes when we vary the number of ZKBoo rounds from 50 to 250, in increments of 50. Increasing the number of rounds increases the bit-strength of the proof, but inadvertently also increases the proof sizes and execution times. The measurements are found in Table 1.

Table 1. Measurement of proof sizes and execution times of $Sign_q$ and $Verify_q$

ZKBoo rounds	50	100	150	200	250
Size of proof (in KBytes)	1,978	3,956	5,934	7,912	9,890
$Sign_q$ execution time (in seconds)	20.9	45.9	72.9	95.1	118.2
$Verify_q$ execution time (in seconds)	19.6	45.0	71.0	93.2	115.7

At first glance, the measured overheads for a 250-bit strength proof show a very large proof of about 10 MB in size and takes almost two minutes to either carry out $Sign_q$ or $Verify_q$. However, when we implement a real-life deployment in Sect. 4, we are able to reduce the impact to the user experience as the proof could be generated asynchronously and stored separately from the certificate.

4 Real-Life Deployment

To study issues related to backward-compatibility and migration to quantum-resistance, we deploy the proposed digital signature scheme into a time-stamp

[1] Source codes can be made available upon request.

server while using an existing (unchanged) Adobe Acrobat Reader DC to request for quantum-resistant time-stamped signed PDFs. The deployment is carried out on a laptop with an Intel I5-8250U 8th Gen machine with 8 CPU cores and 8 GB RAM, running 64-bit Microsoft Windows 10 for both the client and server. The setup is as follows:

- *Time-stamp client.* We use an Adobe Acrobat Reader DC v2021.x. This client already supports ECDSA [3] and is used unmodified.
- *Time-stamp server.* We use an open-source time-stamp server (from https://github.com/kakwa/uts-server) by Pierre-Francois Carpentier. This server is used with codes unmodified.
- *Cryptographic library.* The time-stamp server makes use of OpenSSL v1.1.x to carry out the operations of Certification Authority (CA) issuance of server certificates, as well as to carry out digital signing according to RFC3161 [2]. We modify the version of OpenSSL v1.1.1b to carry out the extended digital signature scheme for both X.509 certificate issuance and time-stamping. An optimization done is to make OpenSSL return the ECDSA signature, while generating the ZKBoo proofs asynchronously. This allows the ECDSA-signed time-stamp to be returned to the client without waiting for the ZKBoo proof to be completely generated. The proofs are thus stored separately from the certificate.
- *Repository.* Since the quantum-resistant 256-round ZKBoo proofs for the certificates and time-stamps are 10 MB each, they could not be easily transmitted to the client. Our modified version of OpenSSL will write the proofs into Dropbox (www.dropbox.com), while embedding the URL link in the signed X.509 certificate or the PKCS#7 time-stamp that is returned to the calling program.

4.1 Deployment Summary

Figure 1 describes the use-cases of the real-life implementation that is tested.

In the setup phase (done once), OpenSSL is used to generate the keys and certify for both the root CA certificate and time-stamp server certificate. We adopt a simple certificate hierarchy where the root CA will certify the server certificate without the need for an intermediate CA (see Fig. 2). Both certificates include the link under the X.509 Authority-Information-Access extension to point to the quantum-resistant proof in Dropbox. The root CA certificate is imported into the Adobe Acrobat to establish the root-to-trust.

In the RFC3161 phase, PDF documents can be timestamped be initiating the request from the Adobe Acrobat which contacts the Time-stamp Server and receives an ECDSA-signed PKCS#7 time-stamp. The time-stamp signature proof is similarly stored in Dropbox with the URL link embedded in the time-stamp. This time-stamp can be verified by the Adobe Acrobat (see Fig. 3) and saved in the PDF. Note that the unmodified Adobe Acrobat only verifies the ECDSA-signed time-stamp and certificate chain and not the ZKBoo proof, resulting in no changes in wait-time experienced by the end-user.

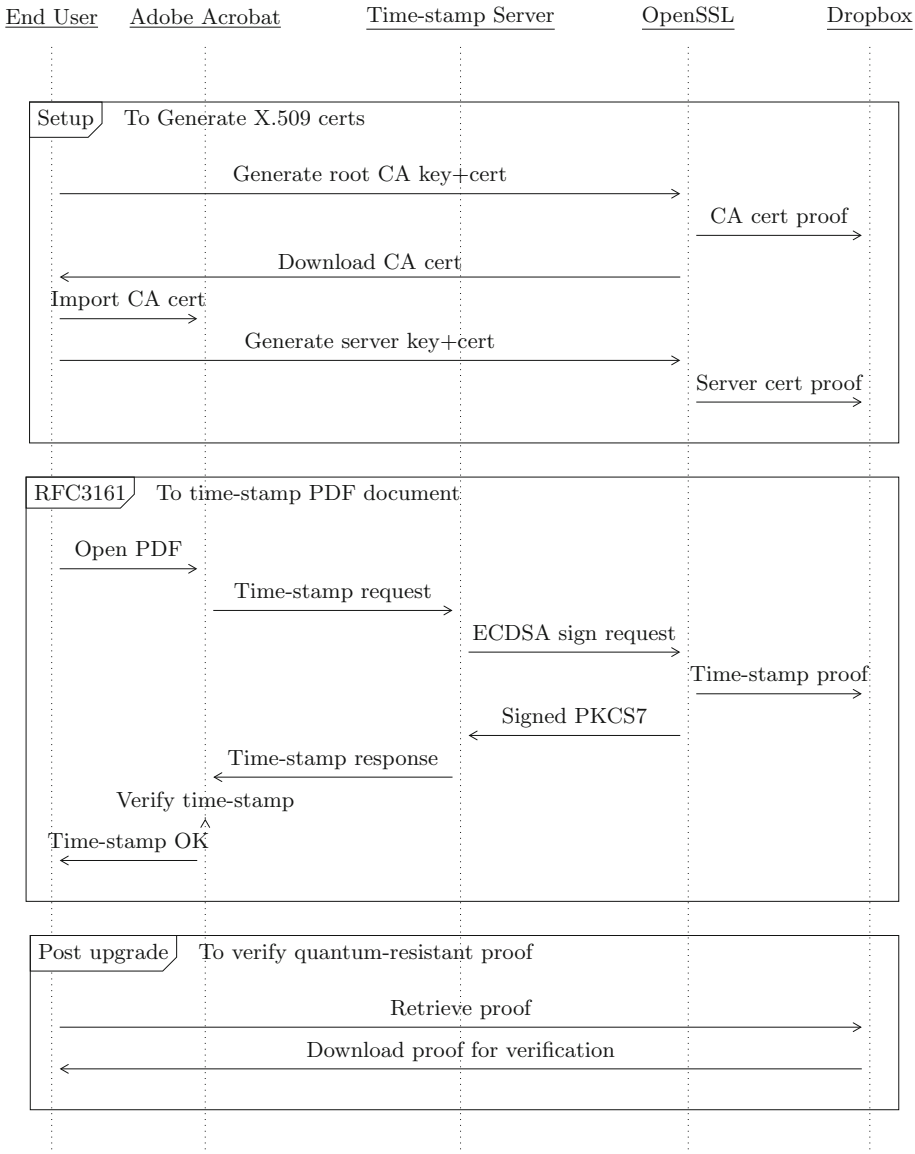

Fig. 1. Real-life deployment of quantum-resistant time-stamp service.

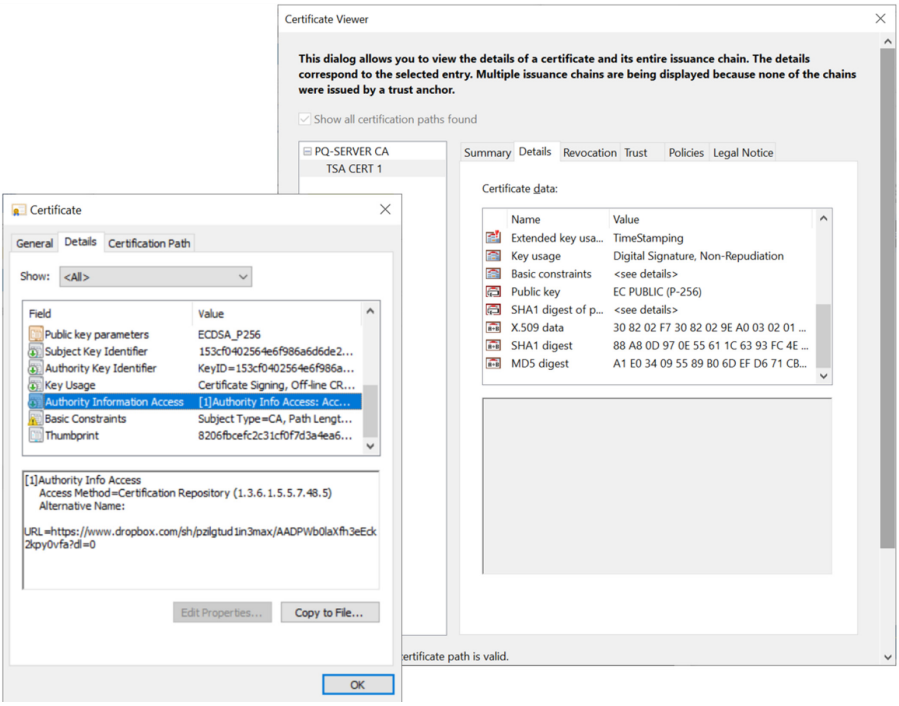

Fig. 2. Root CA and time-stamp server certificates.

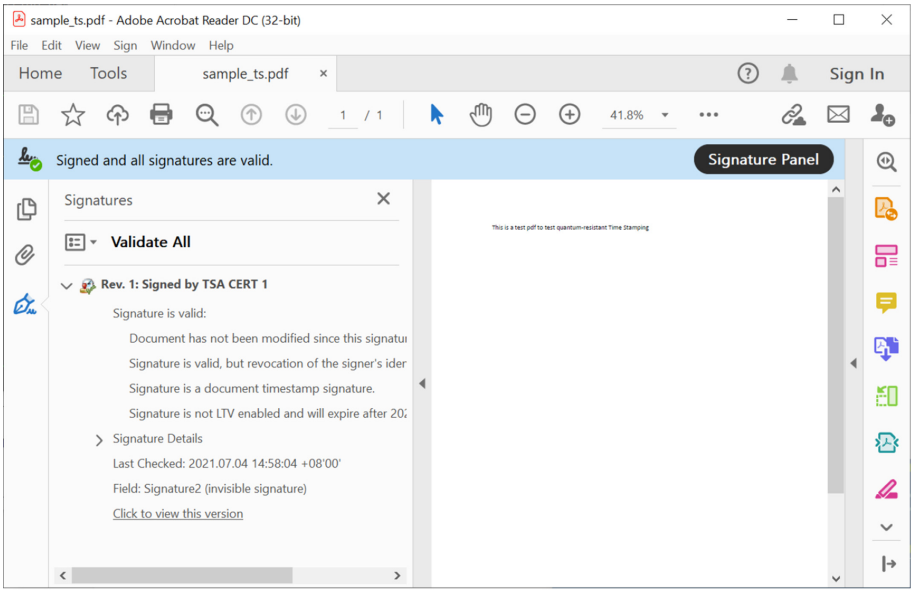

Fig. 3. Time-stamp verification by Adobe Acrobat unaffected by extension.

In the Post upgrade phase, any verifying party capable of running $Verify_q$ can follow the link found in the certificates/signature block to download the quantum-resistant proofs for complete signature verification as per Algorithm 3.

4.2 Exploring Migration

To understand the impact to systems which are gradually migrating to the quantum-resistant digital signature scheme, we list the different outcomes in Table 2 for the signing and verifying parties at different stages of migration.

Table 2. Outcomes for parties at different stages of migration

		Verifying party	
		Definition 1[a]	Definition 2[b]
Signing party	Definition 1[a]	This is the "as-is" scenario. Signed documents are vulnerable to forgery in the post-quantum era.	Signed documents do not include the quantum-resistant proof. Verifying parties have the choice to either reject the document or use $Verify$ to check the signature while informing the signing party to perform the migration
	Definition 2[b]	This is the appropriate "Step 1" of the migration process. Signed documents include the quantum-resistant proof. Signing parties do not need to wait for verifying parties to migrate before carrying out this step. Verifying parties continue to verify the signature while ignoring the quantum-resistant proof	This is the "to-be" state, and the appropriate "Step 2" of the migration process. After the signing parties have migrated and started sending signed documents that include the quantum-resistant proof, the verifying parties can update their systems to verify the signature and proof to complete the migration process

[a]Not yet migrated. Still using Definition 1's $KeyGen$, $Sign$ and $Verify$.
[b]Migration done. Upgraded using Definition 2's $KeyGen_q$, $Sign_q$ and $Verify_q$.

The appropriate migration strategy to layer in quantum-resistance would therefore be to firstly upgrade the signing parties to include the quantum-resistant proof with the signature, before upgrading the verifying parties to be able to verify the proofs. For verifying parties who choose to upgrade early, it is recommended that they include Definition 1's $Verify$ function to maintain compatibility with signing parties who may not have upgraded yet.

5 Related Work

The instinctive approach to make digital signatures quantum-secure is to use a replacement or an additional quantum-secure algorithm. At this point of writing, NIST's post-quantum standardization exercise [29,30] has identified two lattice-based algorithms, Dilithium and Falcon, and one multivariate-based algorithm, Rainbow, as the three finalist digital signature algorithms. Three alternative algorithms, namely multivariate-based GeMSS, zero-knowledge-based Picnic and stateless hash-based SPHINCS$^+$, have been shortlisted but will undergo further

evaluation beyond the year 2024 deadline. Ideally, a "drop-in replacement" in the form of a software library or hardware security module, would be used to swap out or augment RSA/DSA/ECDSA with the new algorithm being standardized. But since each of these algorithms have unique resource, performance and platform considerations [33,36,37], coupled with different key ceremony processes and protocols, it is more likely that a migration playbook [4] needs to be designed and carried out.

Another approach is to use a backup key that can override the regular signing key in the event of compromise. In Chaum et al. [11], the authors propose the use a quantum-resistant stateful hash-based W-OTS$^+$ backup key which is created during the key generation process, and can be used as a fall-back procedure in the event the original key is compromised or lost. While such backup digital signing key approaches can work as an account-recovery mechanism for authentication-related protocols, they are not suitable for routine non-interactive digital signing use-cases where longer-term non-repudiation protection of data is required.

Specific to the time-stamping use-case, the use of a sequence of hashes, chaining them in either a forward or backward direction, is a well-researched approach to provide long-term, possibly quantum-secure, time-stamping. Digital time-stamping by linking the sequence of documents to be time-stamped through a linear hash-chain is first proposed by Haber and Stornetta in 1991 [23]. This is improved by Bayer et al. [5] to use Merkle trees [28] instead of a linear hash-chain. Fast forward to the present where blockchains such as Ethereum already support time-stamping smart contracts [16,31]. Abadi et al. [1] further provided a decentralized time-stamp protocol on blockchains that can prevent pre/post-dating. As these techniques typically rely on a public verifiable chain to determine a specific time of occurrence, they are not applicable as a quantum-resistant mechanism to protect digital signatures in general. Public blockchains also face privacy-related concerns since the number of transactions performed and the timings that they were transacted are publicly available.

6 Conclusion

In this paper, we have taken a different approach in implementing post-quantum digital signing. Instead of replacing or adding on a different quantum-secure digital signing algorithm, we have shown that it is possible to continue to use the classical RSA, DSA or ECDSA digital signing algorithms while achieving longer-term quantum resistance. This is achieved by layering in a zero-knowledge proof of knowledge of the pre-image of the private key in addition to the digital signature. With our approach, digital signature implementations wanting to move ahead in quantum readiness continue to maintain backward-compatibility to existing applications. This is important since different systems may have different timelines and schedules on when the migration to quantum readiness happens, and our approach is able to ensure the seamless operations between upgraded and non-upgraded applications.

But the work is not yet complete. The current implementation is neither computational-efficient nor space-efficient. We envision future work in:

- *Optimizing the zero-knowledge proof.* Both the size of the zero-knowledge proof and execution times for $Sign_q$ and $Verify_q$ can be significantly improved. There are other MPC-in-the-head based zero-knowledge proofs such as ZKB++ [9], TurboIKOS [22] which achieve better performance and/or space efficiencies. Our implementation can be updated to use them.
- *Increasing the spectrum of applications.* Beyond time-stamping PDF documents, there are other applications such as blockchain, Secure Email, and Transport Layer Security that can use this approach to layer in quantum resistance.
- *Updating standards.* In our implementation, the X.509 Authority Information Access extension is used to store the link for the zero-knowledge proof. This may clash with other applications already using this extension and hence may not be the most appropriate use of the extension. A standardized extension can be established for storing such zero-knowledge proofs.

Acknowledgement. This project is supported by the Ministry of Education, Singapore, under its MOE AcRF Tier 2 grant (MOE2018-T2-1-111).

References

1. Abadi, A., Ciampi, M., Kiayias, A., Zikas, V.: Timed signatures and zero-knowledge proofs—timestamping in the blockchain era—. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 2020. LNCS, vol. 12146, pp. 335–354. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57808-4_17
2. Adams, C., Cain, P., Pinkas, D., Zuccherato, R.: RFC 3161: Internet x. 509 public key infrastructure time-stamp protocol (TSP) (2001)
3. Adobe: Adobe DC Digital Signatures Guide - Supported Standards (2018). https://www.adobe.com/devnet-docs/acrobatetk/tools/DigSigDC/standards.html. Accessed Apr 2021
4. Barker, W., Polk, W., Souppaya, M.: Getting ready for post-quantum cryptography: explore challenges associated with adoption and use of post-quantum cryptographic algorithms. The Publications of NIST Cyber Security White Paper (DRAFT), CSRC, NIST, GOV 26 (2020)
5. Bayer, D., Haber, S., Stornetta, W.S.: Improving the efficiency and reliability of digital time-stamping. In: Capocelli, R., De Santis, A., Vaccaro, U. (eds.) Sequences II, pp. 329–334. Springer, New York (1993). https://doi.org/10.1007/978-1-4613-9323-8_24
6. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptology ePrint Archive 2018/46 (2018)
7. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.: Strengths and weaknesses of quantum computing. SIAM J. Comput. **26**(5), 1510–1523 (1997)
8. Certicom: SEC 2: Recommended elliptic curve domain parameters. Technical Report SEC2-Version-1.0, Certicom Research, Mississauga, ON, Canada (2000)
9. Chase, M., et al.: The picnic digital signature algorithm: update for round 2 (2019)
10. Chase, M., 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, pp. 1825–1842. ACM (2017)

11. Chaum, D., Larangeira, M., Yaksetig, M., Carter, W.: W-OTS$^+$ up my sleeve! a hidden secure fallback for cryptocurrency wallets. In: Sako, K., Tippenhauer, N.O. (eds.) ACNS 2021. LNCS, vol. 12726, pp. 195–219. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78372-3_8
12. Chen, L.: Cryptography standards in quantum time: new wine in old wineskin? IEEE Secur. Priv. **15**(4), 51 (2017)
13. Chen, L., et al.: NISTIR 8105: Report on post-quantum cryptography. US Department of Commerce, National Institute of Standards and Technology (2016)
14. Cooper, D.A., Apon, D.C., Dang, Q.H., Davidson, M.S., Dworkin, M.J., Miller, C.A.: Recommendation for stateful hash-based signature schemes. NIST Special Publication 800-208 (2020)
15. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
16. Estevam, G., Palma, L.M., Silva, L.R., Martina, J.E., Vigil, M.: Accurate and decentralized timestamping using smart contracts on the Ethereum blockchain. Inf. Process. Manag. **58**(3), 102471 (2021)
17. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
18. FIPS PUB: 180-4. Secure Hash Standard (SHS). Information Technology Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg (2015)
19. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: faster zero-knowledge for Boolean circuits. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 1069–1083 (2016)
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
21. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. **79**(2), 325 (1997)
22. Gvili, Y., Ha, J., Scheffler, S., Varia, M., Yang, Z., Zhang, X.: TurboIKOS: improved non-interactive zero knowledge and post-quantum signatures. In: Sako, K., Tippenhauer, N.O. (eds.) ACNS 2021. LNCS, vol. 12727, pp. 365–395. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78375-4_15
23. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 437–455. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_32
24. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, pp. 21–30. ACM (2007)
25. Jones, J.P., Sato, D., Wada, H., Wiens, D.: Diophantine representation of the set of prime numbers. Am. Math. Mon. **83**(6), 449–464 (1976)
26. Kerry, C., Gallagher, P.: FIPS PUB 186-4: Digital signature standard (DSS). Federal Information Processing Standards Publication, National Institute of Standards und Technology (2013)
27. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
28. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_40

29. Moody, D.: NIST Status Update on the 3rd Round (2021). https://csrc.nist.gov/CSRC/media/Presentations/status-update-on-the-3rd-round/images-media/session-1-moody-nist-round-3-update.pdf. Accessed July 2021
30. NIST: Post-Quantum Cryptography: Round 3 Submissions (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions. Accessed July 2021
31. Pastor, M., dela Eva, R.: TimeStamp Smart Contract (2021). https://ec.europa.eu/cefdigital/wiki/display/EBSIDOC/TimeStamp+Smart+Contract. Accessed July 2021
32. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. arXiv preprint quant-ph/0301141 (2003)
33. Raavi, M., Wuthier, S., Chandramouli, P., Balytskyi, Y., Zhou, X., Chang, S.-Y.: Security comparisons and performance analyses of post-quantum signature algorithms. In: Sako, K., Tippenhauer, N.O. (eds.) ACNS 2021. LNCS, vol. 12727, pp. 424–447. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78375-4_17
34. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
35. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
36. Sikeridis, D., Kampanakis, P., Devetsikiotis, M.: Post-quantum authentication in TLS 1.3: a performance study. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 23–26 February 2020. The Internet Society (2020)
37. Tan, T.G., Szalachowski, P., Zhou, J.: SoK: challenges of post-quantum digital signing in real-world applications. Cryptology ePrint Archive, Report 2019/1374 (2019). https://eprint.iacr.org/2019/1374
38. Unruh, D.: Collapse-binding quantum commitments without random oracles. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 166–195. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_6