Towards Quantum Resistant Trusted Computing: Architectures for Post-Quantum Integrity Verification Techniques

Towards Quantum-Resistant Trusted Computing: Architectures for Post-Quantum Integrity Verification Techniques

Grazia D'Onghia
Politecnico di Torino
Dip. di Automatica e Informatica
Torino, Italy
grazia.donghia@polito.it

Antonio Lioy
Politecnico di Torino
Dip. di Automatica e Informatica
Torino, Italy
antonio.lioy@polito.it

***Abstract*—Trust is the core building block of secure systems, and it is enforced through methods to ensure that a specific system is properly configured and works as expected. In this context, a Root of Trust (RoT) establishes a trusted environment, where both data and code are authenticated via a digital signature based on asymmetric cryptography, which is vulnerable to the threat posed by Quantum Computers (QCs). Firmware, being the first layer of trusted software, faces unique risks due to its longevity and difficult update. The transition of firmware protection to Post-Quantum Cryptography (PQC) is urgent, since it reduces the risk derived from exposing all computing and network devices to quantum-based attacks. This paper offers an analysis of the most common trust techniques and their roadmap towards a Post-Quantum (PQ) world, by investigating the current status of PQC and the challenges posed by such algorithms in existing Trusted Computing (TC) solutions from an integration perspective. Furthermore, this paper proposes an architecture for TC techniques enhanced with PQC, addressing the imperative for immediate adoption of quantum-resistant algorithms.**

I. Introduction

In digital systems, the concept of *trust* is related to Trusted Computing Base (TCB) and RoT. A TCB is the totality of protection mechanisms within a computer system (including hardware, firmware, and software), and their combination is responsible for enforcing a security policy [1], while a RoT is a component that performs one or more security-specific functions and is always assumed to behave in the expected manner [2]. TC first became a reality with the Trusted Platform Module (TPM), which now can be found in millions of laptops and electronic devices around the world. Thanks to this component, the building blocks that allow one component in a computer network to trust all other hardware and software related pieces can be established [3]. However, since TC is based on authentication, that is, digital signatures using public key cryptography, it is threatened by the advent of Cryptographically Relevant Quantum Computers (CRQCs), as demonstrated by Shor's algorithm [4], capable of factoring large prime numbers in polynomial time. The urgency of migrating to PQC is enhanced by the "Store now, decrypt later" attack. The latter exploits the asymmetry between the time it takes to develop a CRQC and the time required to migrate existing systems to PQ solutions. This dynamic is described in Mosca's inequality [5], which warns that systems lacking quantum resistance today risk catastrophic compromise tomorrow. This threat is particularly acute for firmware protection, where digital signatures must ensure integrity over long device lifecycles. The US National Security Agency (NSA) underscores this urgency in its Commercial National Security Algorithm (CNSA) 2.0 Suite [6], explicitly prioritizing post-quantum firmware signing as a critical defence against future quantum adversaries. Once compromised, firmware vulnerabilities may undermine entire ecosystems, from Internet of Things (IoT) sensors to cloud infrastructure. This paper directly addresses this imperative, offering actionable recommendations for integrating PQ algorithms into firmware signing and attestation workflows, aligning with CNSA 2.0's call for immediate transitions. The first essential step in this transition exercise is to evaluate the already standardized PQC algorithms proposed by the NIST [7]. The purpose of this evaluation is to understand the most suitable algorithm for the specific use case of TC and Integrity Verification. Since PQC relies on different mathematical problems and uses keys much larger than current signatures, compatibility must be evaluated with current instruction sets and network protocols. The contribution of this paper consists in combining the previous recommendations with our requirements to evaluate the best approach towards PQ Integrity Verification techniques. The effort in the PQ transition relies in evaluating the overhead introduced by PQ algorithms and choosing the best ones for two use cases within a Remote Attestation framework. This paper is structured as follows: section II contains an overview of the main Integrity Verification techniques, section III summarizes the status of PQC standardization with a comparison among standardized algorithms, and section IV contributes to the PQ transition by providing recommendations and requirements on the integration of PQC within Integrity Verification techniques. Finally, Section V contains our proposed architecture for a system that integrates PQC in Integrity Verification, and section VI contains the conclusion and examines future works.

II. Trusted Computing techniques for Firmware and Software protection

Endpoint computers encompass hardware, firmware, drivers, operating systems, and application software, all of which impact the integrity and security of both the devices themselves and the network they are part of. Modern software and firmware protection rely on three core techniques: secure boot, measured boot, and Remote Attestation. Such techniques have become an essential requirement, especially to protect IoT devices, since attacks against them have become widespread. The integrity of an IoT device system includes load-time integrity (secure and measured boot) and runtime integrity (Remote Attestation) [8]. Consequently, there is a difference between integrity verification enforced on the platform (such as secure boot) and Remote Attestation, which allows external entities to verify the integrity status of a device. Integrity Verification enforced on the platform is realized with the Root of Trust for Measurement (RTM), founded in the Core Root of Trust for Measurement (CRTM), which is an immutable portion of the platform initialization code responsible to verify the authenticity of the next entity before transferring control to it. Meanwhile, Remote Attestation requires the platform to provide three RoTs: RTM, Root of Trust for Storage (RTS), and Root of Trust for Reporting (RTR).

A. Secure Boot

Secure boot ensures that a device starts only with trusted software. This is achieved by verifying the digital signature created by the manufacturer on the boot components. In this way, a Chain of Trust (CoT) is established to ensure the integrity of the load time of each boot partition. The CoT is based on the concept of verifying the next boot image: the image of the former boot partition verifies the image of the next boot partition, and so on until all the boot partitions are successfully loaded. The essential aspect of secure boot is that even if a single signature verification fails, the whole boot process is halted. Transitioning to a quantum-resistant secure boot involves supporting PQC in the boot sequence at the earliest possible level, to protect the whole boot chain from quantum attacks. Secure boot also requires a careful evaluation of resource requirements and performance, as the PQ signature algorithms have to be efficient in the verification operation to keep boot time and system latency low.

B. Measured Boot

Measured boot measures the elements in the CoT, from power on until the operating system is fully loaded. During the boot flow, each critical system component (e.g., firmware, bootloaders, kernel) is measured by the previous one before it gains control of the platform. The measurement is typically a cryptographic digest computed with a secure hash function, such as SHA-256. The measured boot chain of trust starts with a CRTM, which is typically implemented with a secure and immutable component, contained in Read-Only Memory (ROM). The state of a system changes as programs run with particular configurations. Measured boot

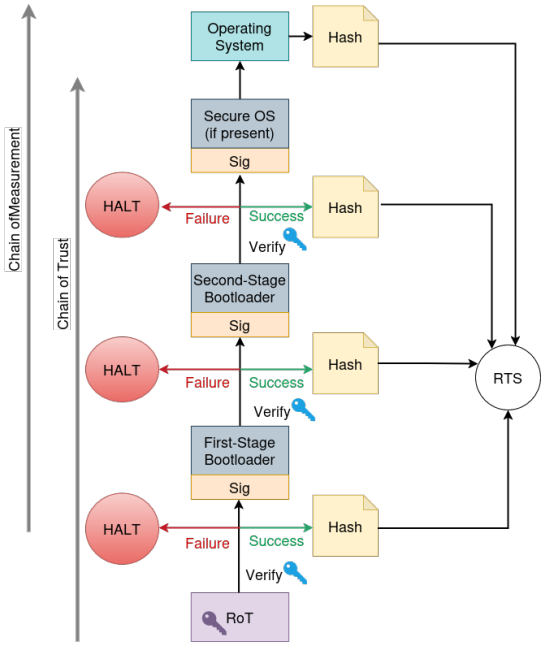

Fig. 1: General view of Secure boot and Measured boot.

accumulates a list of measurements for each program executed, but does not perform any enforcement like secure boot, which instead halts the system if any attempt is made to execute a program that is not on an approved list [9]. However, both systems involve measuring programs before executing them. The measurements acquired during the boot stages must be recorded in a secure environment, which usually is the TPM, either physical or firmware running inside a Trusted Execution Environment (TEE). Afterwards, the recorded measurements will be available for use in the Remote Attestation protocol. Figure 1 represents the processes of creating a CoT (during secure boot) and a chain of measurement (during the measured boot). The main difference between them is that secure boot blocks the execution of the boot process by launching a *halt* if a verification fails, while measured boot records the cryptographic hashes of code portions (the measurements), and stores them in the RTS. In this way, secure boot creates a chain of trust, while measured boot creates a chain of measurements.

C. Remote Attestation

Remote Attestation is a protocol used to verify the integrity and trustworthiness of a remote computing system. Remote Attestation verifies the integrity and trustworthiness of a remote system (the Attester), which provides reliable information about itself (called evidence) to enable a remote peer (the Relying Party) to decide whether or not to consider that Attester a trustworthy peer. Additionally, another component called Verifier appraises the evidence via appraisal policies and creates the attestation result to support Relying Parties in their decision process [10]. In some architectures, the Relying

Party and the Verifier may be merged in a single actor. Remote Attestation is a challenge-response protocol in which the Verifier sends a challenge (a nonce) to the Attester. The nonce is used to avoid replay attacks, in which the attacker would send back to the Verifier an old attestation evidence, created before the system is corrupted. When the attesting system receives the attestation challenge from the Verifier, it generates an attestation evidence, called Quote, which serves as proof that the device's current state (including its firmware, software, and configuration) is trustworthy and has not been tampered with. The key elements of a Quote are the measurements performed on the system components, the nonce received from the Verifier and the signature made on the measurement data and the nonce to ensure integrity and authenticity. In order to enable runtime attestation of a computational node, the Integrity Measurement Architecture (IMA) module, provided by the Linux kernel, must be appropriately configured with a policy. Finally, Attester and Verifier must communicate through secure protocols, such as Transport Layer Security (TLS) in order to protect the privacy of integrity measurements in transit on the network.

III. Post-Quantum Cryptography

As of 2024, NIST has finalized four algorithms for standardization: Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) [11], Module-Lattice-Based Digital Signature (ML-DSA) (was CRYSTALS-Dilithium), Stateless Hash-Based Digital Signature (SLH-DSA) (was SPHINCS+), and FFT over NTRU-Lattice-Based Digital Signature Algorithm (FN-DSA) (was FALCON), whose standard has not been published yet. Due to the significant differences in implementation, related mathematical problems, memory consumption, and performance, it is essential to differentiate standardized algorithms in order to find the most suitable ones for specific use cases. Unlike Key Encapsulation Mechanisms (KEMs), digital signatures require long-term security guarantees because signatures can be harvested and attacked retroactively once large-scale quantum computers exist [12]. This procedure represents the first engineering step for the adoption of PQC in existing cryptosystems, and it is as relevant as the mathematical study and implementation of the algorithms themselves. Standardization is just the first step of actual deployment in existing cryptosystems, and the biggest challenge is including PQC in all communication layers and all devices, from clients to service back-ends [13].

Table I contains a summary of the standardized PQ digital signature algorithms with a focus on practical and integration aspects. Lattice-Based algorithms leverage complex mathematical problems such as Module Learning With Errors (used in ML-DSA) and achieve compact key sizes and an overall efficient verification. ML-DSA supports NIST security levels 2 (ML-DSA-44), 3 (ML-DSA-65), and 5 (ML-DSA-87), offers a balance between speed and security. With public keys in the range 1.3-2.6 kB and signatures 2.4-4.6 kB, it is well-suited for every application, but it is recommended for Public Key Infrastructure (PKI) and Internet protocols. Moreover, Duarte de Menese et al. [14] recently provided an optimization to reduce its computational overhead by shrinking its memory footprint. However, ML-DSA is susceptible to side-channel attacks, thus requiring careful implementation hardening. FN-DSA, implemented with NIST levels 1 (FN-DSA-512) and 5 (FN-DSA-1024), distinguishes itself with exceptionally small signatures (0.6–1.3 kB), making it ideal for resource-constrained environments such as IoT devices. Nevertheless, its reliance on 53-bit Floating Point Unit (FPU) [15] limits adoption in low-end embedded systems lacking hardware FPU.

Hash-Based (HB) signatures are uniquely suited for integrity verification due to their reliance on quantum-secure hash functions rather than structured mathematical problems. These algorithms prioritize long-term security through collision-resistant hash functions, though they introduce a significant overhead in performance. HB schemes are recommended mostly for safeguarding integrity-sensitive systems, particularly in domains that require high security such as firmware signing, secure boot [6], and Certification Authoritys (CAs) [16]. Among these, SLH-DSA stands out for its stateless design, which does not involve state management. The stateless nature of SLH-DSA simplifies deployment in decentralized architectures, but its operational costs limit adoption to niche, high-security domains. SLH-DSA's large signature sizes (7.8–49.8 kB) and slow signature generation make its adoption for real-time applications a big challenge. In contrast, stateful HB schemes like XMSS and LMS produce smaller signatures (2.5–12 kB) and adhere to RFC standards [17]. These algorithms are optimized for use cases such as secure boot chains and firmware signing. LMS, in particular, demonstrates strong performance on embedded systems [18], but both XMSS and LMS require robust state management to mitigate key-reuse risks, thus requiring solutions such as TEEs to manage the state. Despite these operational complexities, stateful HB schemes are interesting options for the PQ transition of TC architectures, especially in resource-constrained devices.

Both lattice-based and HB algorithms can be implemented in TC, but HB algorithms rely on the well-established security of the underlying hash functions, thus being more suitable for such a high security purpose. Despite providing the same NIST security levels, it is worth noting that the security of lattice-based algorithms has been investigated less since they are based on novel mathematical problems.

IV. Challenges of PQC integration in integrity verification techniques

Migrating Integrity Verification techniques to PQC is a complex, yet essential transition exercise that demands a thorough revision of key management practices, cryptographic libraries, and protocol standards used into existing frameworks. This section analyzes system requirements and designs building blocks needed to carry out transition of Integrity Verification techniques to PQC on different device categories: embedded systems and network platforms.

TABLE I: Qualitative overview of Lattice-Based and Hash-Based Post-Quantum Algorithms

Algorithm	NIST security Levels	Key/Signature Sizes	Strengths	Weaknesses	Best Use Case
Lattice-Based					
ML-DSA	2, 3, 5	Pub: 1,312–2,592 B Priv: 2,560–4,896 B Sig: 2,420–4,627 B	- Balance of speed and security [12] - Optimizations made [14]	- Large signatures - Vulnerable to side-channel attacks [19]	- PKI - Internet applications
FN-DSA	1, 5	Pub: 897–1793 B Priv: 1,281–2,305 B Sig: 666–1,280 B	- Smallest signatures - Efficient verification	- Requires 53-bit FPU	- Resource-constrained environments - IoT devices
Hash-Based					
SLH-DSA	1, 3, 5	Pub: 32–64 B Priv: 64–128 B Sig: 7,856–49,856 B	- Stateless - Conservative security	- Massive signatures - Slow signing/verification	- Long-term integrity (e.g., firmware, logs) - High security domains (e.g., CA) [16]
XMSS	1, 3, 5	Pub: 32–64 B Priv: 1,088–2,560 B Sig: 2,500–12,000 B	- Stateful but smaller signatures than SLH-DSA - RFC 8391 standardized [17]	Requires secure state management	- Secure boot CoT [20] - Firmware signing [6]
LMS	1, 3, 5	Pub: 32–64 B Priv: 1,088–2,560 B Sig: 2,500–12,000 B	- Stateful but smaller signatures than SLH-DSA - Best performance among HB on embedded systems [18]	- Requires secure state management	Secure boot CoT [21]

A. Secure Boot

Making secure boot quantum-resistant requires first PQC integration at the firmware level, to ensure that the initial code executed during the boot process is authenticated using a PQ signature. Some key factors must be considered when choosing the optimal digital signature algorithm to use for PQ secure boot. First, PQ digital signatures used for secure boot require a high security strength and algorithm maturity, as they should remain valid for a long period of time. The security of HB schemes relies solely on the properties of the underlying hash functions, which are well-studied and understood in cryptography, providing a solid foundation for their security compared to newer approaches, such as lattice-based ones [22]. This consideration leads to preferring HB schemes in the implementation of PQ secure boot. Among HB schemes, NIST selected SLH-DSA for standardization, while the IETF developed RFCs for two Stateful HB schemes, LMS [23] and XMSS [17]. Furthermore, it is crucial to ensure that the integration of PQ algorithms does not introduce new vulnerabilities in the secure boot process. Stateful HB algorithms (LMS, XMSS) require secure and reliable mechanisms for state storage and updates to ensure their security and correct operation. Therefore, their adoption would add complexity. However, when these algorithms are used to implement secure boot, they do not increase the implementation complexity in the device, as the management and updating of the state have to be carried out by the entity that signs the software, which is external to the device. Another critical aspect of the secure boot process is the management of cryptographic keys. PQ key management involves first generating PQ public and private keys by the firmware and software provider. The private key is used to sign the firmware and software, so it has to be stored securely by the software provider and never shared. The public key is instead used during the device boot to verify the authenticity and integrity of the boot components, so it has to be distributed and embedded in the device's firmware, or in its RoT. If a private key is compromised, a revocation mechanism must be in place to invalidate the old key and prevent its use. This often involves updating the firmware with a new public key. Therefore, PQ secure boot should support a secure firmware update mechanism that includes updating the embedded public key. When choosing the PQ signature algorithm, it is necessary to take into account that the adoption of stateful HB schemes would add complexity to private key management, because the signer must keep track of the used and remaining private key components. This state must be securely stored and updated after each signature generation to prevent reuse of the same components, which would compromise security. There are several works that demonstrate the feasibility of PQ secure boot through the adoption of HB algorithms.[24] presents a comprehensive evaluation of signature verification of the standardized NIST PQ algorithms, showing that HB algorithms have the best performance. XMSS algorithm is adopted for secure boot in [20], offering great performance and reduced signature sizes. Other HB schemes such as LMS and SPHINCS+ are evaluated in [21].

B. Measured Boot

Implementing a PQ measured boot requires adopting quantum-resistant hash functions to generate cryptographic hashes of code and configurations. The impact of a CRQC on hash functions is less severe than the impact on digital signatures, but still significant. The security strength of a hash function against quantum attacks is primarily evaluated based on Grover's algorithm [25], which can speed up the brute-force search for pre-image attacks, effectively reducing the security level of the hash function by half [26]. The best known quantum attack does not improve significantly over the classical birthday attack, which halves the security strength [19]. Therefore, to implement a PQ measured boot, hash

functions like SHA-384, SHA-512, or SHA-3-512 should be used, ensuring sufficient collision resistance against quantum threats. Then, the recorded quantum-resistant measurements are available to be used in the PQ remote attestation protocol.

C. Remote Attestation

Remote attestation requires some considerations about PQC integration, specifically for the hash and signatures algorithms employed. To enable measurement of files accessed at system runtime with quantum-resistant hash algorithms, the IMA module has to be configured to use these algorithms (e.g. SHA-512, available since kernel version 3.13, or SHA-3-512 available since version 6.7). Furthermore, to protect the integrity of the IMA Log file with SHA-512 or SHA-3-512, a TPM must be present in the system and it must be configured with a Platform Configuration Register (PCR) bank associated with SHA-512 or SHA-3-512. In the case of PQ remote attestation, the selected signature algorithm should be efficient in all its operations, especially for signature generation and verification. The efficiency of these operations is crucial for quickly detecting compromises and maintaining overall system performance: the faster the attestation process, the shorter the window of opportunity for attackers to exploit a compromised system. An adoption of PQ algorithms in Remote Attestation can be found in [27], where Dilithium2 and FALCON are evaluated against ECDSA. In this work, FALCON represents an optimal choice for PQ remote attestation, because it provides the highest efficiency in signature generation and verification. Since different evaluations can be made for remote attestation based on the use case, enabling algorithm agility for the attestation evidence signature is a desirable feature.

V. Proposed Architecture

The architecture proposed here finalizes the investigation that has been done about PQC and its integration into firmware and software protection techniques. However, this is the first step in the complete transition of cryptosystems, which includes the full hardware integration of PQ standardized algorithms. At its core, the architecture operates on two parallel fronts, each tailored to a distinct deployment environment. Figure 2 represents the high level architecture of the framework.

For modern ARM-based embedded systems, trust is anchored in a PQ Firmware TPM (fTPM) running within ARM TrustZone. This fTPM includes PQ signatures to sign the attestation Quote , ensuring quantum-resistant integrity verification. This solution addresses an use case where the PQ transition must be done on a system without a physical TPM, thus using a firmware TPM.

On the other side, for x86 systems reliant on physical TPM, the architecture adopts a hybrid model that combines classical ECDSA signatures with PQ wrappers at the kernel level. After receiving the attestation request from the Verifier, the Attester asks the physical TPM to generate the Quote, which is first signed with ECDSA and then wrapped with a PQ signature by an extension in the TPM driver, obtaining a hybrid Quote, thus adapting to this first phase of the transition. Meanwhile,

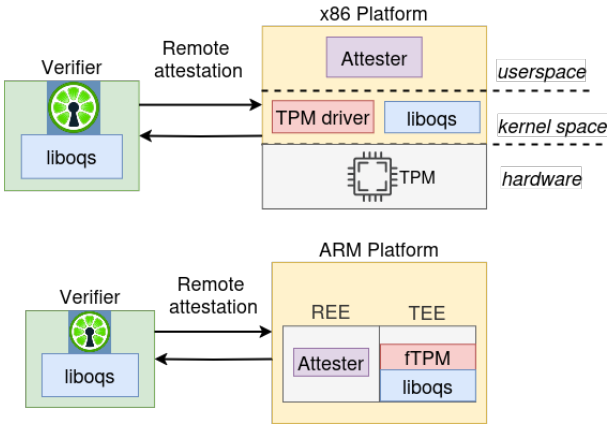

Fig. 2: High level architecture of the proposed solution.

the Verifier enhances the Keylime framework [28] with PQC support via liboqs library [29], in order to understand the PQ Quote received from the Attester, whether it is ARM or x86, and to be able to perform signature verification.

The key difference in the two flavours is the generation of the Quote.

For the ARM platform it is important to highlight the difference between the Rich Execution Environment (REE) (namely, the Normal World in ARM TrustZone) and the TEE.

On the other side, the meaningful separation that is drawn in the x86 platform is between userspace (where regular applications run), kernel space (with kernel drivers that interact directly with the hardware), and the hardware itself, where the physical TPM is located. Acknowledging the phased nature of PQ adoption, the architecture prioritizes interoperability. Hybrid workflows allow legacy systems to coexist with PQ-enhanced components.

On ARM platforms, the fTPM demonstrates how constrained devices can adopt PQ primitives in integrity verification without needing a physical TPM. On x86 systems, kernel-level PQ wrappers provide an assurance for environments where hardware upgrades are impractical. Together, these strategies close critical gaps in today's trust chains while preparing the ground for fully PQ-native ecosystems. This solution is only at design stage, and the evaluation of the implementation and performance of the adopted algorithms will be done in another research activity. However, when dealing with runtime operations such as Remote Attestation, security strength must be paired with performance, especially in signature generation (involved in the creation of the Quote), therefore the optimal algorithm for our solution is ML-DSA.

VI. Conclusion and Future Work

Quantum computers will force a paradigm shift in how we safeguard digital system, especially firmware and software ecosystems which need long-term security. By proposing a layered architecture that integrates PQC across the different steps of integrity verification, this paper addresses a critical gap in today's cybersecurity landscape: the lack of end-to-end

quantum-resistant trust chains. Current architectures often treat secure boot, measured boot, and Remote Attestation as isolated processes, creating fragmented trust boundaries that could be eventually exploited by quantum adversaries. This work redefines integrity verification as a cohesive, layered framework where PQC is included into every stage of trust propagation, thus making a foundational step toward cryptographic continuity, where systems adapt dynamically to emerging threats. This design bridges the divide between legacy systems and emerging PQ standards, offering a dual-path strategy, for modern ARM-based platforms and legacy x86 systems, that balances innovation with pragmatic transition needs. While this paper outlines an important draft, the journey to practical, scalable PQ adoption has only begun. Future works include rigorous testing and evaluation of the proposed architectures to quantify their real-world effect. Future research should explore the development of lightweight PQ primitives optimized for firmware protection, such as HB signatures with smaller footprints or lattice-based schemes with reduced computational overhead. As quantum computing transitions from theory to reality, the lessons of this paper are clear: firmware protection is a building block of global cybersecurity, consequently its PQ transition is inevitable. The designs proposed here are not endpoints but basis for a collective, urgent effort to quantum-resistant trust architectures. Implementation details, empirical validations, and policy frameworks will follow in subsequent work, but the concept is very clear: the time to act is now.

Acknowledgments. This work is part of the QUBIP European project – https://qubip.eu/ – funded by the European Union under the Horizon Europe framework programme (grant agreement no. 101119746).

REFERENCES

[1] P. S. Tasker, W. F. Shadle, J. P. Anderson, and S. B. Lipner, "Trusted Computer System Evaluation Criteria (Orange Book) December", 2001. https://api.semanticscholar.org/CorpusID:18200746

[2] Trusted Computing Group, "TCG Glossary Version 1.1", https://trustedcomputinggroup.org/wp-content/uploads/TCG-Glossary-V1.1-Rev-1.0.pdf

[3] Trusted Computing Group, "Trusted Computing: The essential building blocks to a secure system", https://trustedcomputinggroup.org/trusted-computing-the-essential-building-blocks-to-a-secure-system/

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM Journal on Computing, vol. 26, no. 5, 1997, pp. 1484–1509, doi:10.1137/S0097539795293172

[5] M. Mosca, "Cybersecurity in a Quantum World: will we be ready?", 2015, https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/presentations/session8-mosca-michele.pdf

[6] National Security Agency, "Announcing the Commercial National Security Algorithm Suite 2.0", 2022, https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF

[7] NIST, "NIST to Standardize Encryption Algorithms That Can Resist Attack by Quantum Computers", 2023, https://www.nist.gov/news-events/news/2023/08/nist-standardize-encryption-algorithms-can-resist-attack-quantum-computers

[8] Z. Ling, H. Yan, X. Shao, J. Luo, Y. Xu, B. Pearson, and X. Fu, "Secure boot, trusted boot and remote attestation for arm trustzone-based iot nodes", Journal of Systems Architecture, 2021, p. 102240, doi:10.1016/j.sysarc.2021.102240

[9] B. Parno, J. M. McCune, and A. Perrig, "Bootstrapping Trust in Commodity Computers", IEEE Symposium on Security and Privacy, Oakland (CA, USA), May 2010, pp. 414–429, doi:10.1109/SP.2010.32

[10] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC-9334, January 2023, doi:10.17487/RFC9334

[11] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", NIST FIPS 203, 2024, https://csrc.nist.gov/pubs/fips/203/final

[12] A. Banerjee *et al.*, "Internet draft: Post-quantum cryptography for engineers", 2024, https://datatracker.ietf.org/doc/draft-ietf-pquip-pqc-engineers/06/

[13] P. Muzikant and J. Willemson, "Deploying Post-quantum Algorithms in Existing Applications and Embedded Devices", 3rd International Conference (UbiSec 2023), Exeter (UK), November 1–3, 2023, pp. 147–162, doi:10.1007/978-981-97-1274-8_10

[14] R. Meneses, C. Teixeira, and M. Henriques, "Compact Memory Implementations of the ML-DSA Post-Quantum Digital Signature Algorithm", Anais Estendidos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2024), São José dos Campos (Brazil), September 2024, pp. 233–243, doi:10.5753/sbseg_estendido.2024.243388

[15] M. Pursche, N. Puch, S. N. Peters, and M. P. Heinl, "SoK: The Engineer's Guide to Post-Quantum Cryptography for Embedded Devices", Cryptology ePrint Archive, Paper 2024/1345, 2024, https://eprint.iacr.org/2024/1345

[16] G. D'Onghia, D. G. Berbecaru, and A. Lioy, "Shaping a quantum-resistant future: Strategies for post-quantum pki", 2024 IEEE Symposium on Computers and Communications (ISCC), Paris (France), June 2024, pp. 1–6, doi:10.1109/ISCC61673.2024.10733624

[17] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC-8391, May 2018, doi:10.17487/RFC8391

[18] F. Campos, T. Kohlstadt, S. Reith, and M. Stöttinger, "LMS vs XMSS: Comparison of Stateful Hash-Based Signature Schemes on ARM Cortex-M4", Africacrypt 2020, Cairo (Egypt), July 2020, pp. 258–277, doi:10.1007/978-3-030-51938-4_13

[19] D. J. Bernstein and T. Lange, "Post-quantum cryptography", Nature, vol. 549, 2017, pp. 188–194, doi:10.1038/nature23461

[20] V. B. Y. Kumar, N. Gupta, A. Chattopadhyay, M. Kasper, C. Krauß, and R. Niederhagen, "Post-Quantum Secure Boot", 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020, pp. 1582–1585, doi:10.23919/DATE48585.2020.9116252

[21] P. Kampanakis, P. Panburana, M. Curcio, and C. Shroff, "Post-quantum Hash-Based Signatures for Secure Boot", Silicon Valley Cybersecurity Conference, San Jose (CA, USA), December 17–19, 2020, pp. 71–86, doi:10.1007/978-3-030-72725-3_5

[22] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical Stateless Hash-Based Signatures", Eurocrypt 2015, Sofia (Bulgaria), April 26–30, 2015, pp. 368–397, doi:10.1007/978-3-662-46800-5_15

[23] D. McGrew, M. Curcio, and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC-8554, April 2019, doi:10.17487/RFC8554

[24] M. Agrawal, K. Duraisamy, K. S. Ganesan, S. Gupta, S. Kandele, S. S. Konduru, H. C. Maddipati, K. Raghavendra, R. A. Sahu, and V. Saraswat, "Secure Boot in Post-Quantum Era", INDOCRYPT 2023, Goa (India), December 10–13, 2023, pp. 223–239, doi:10.1007/978-3-031-56235-8_11

[25] L. K. Grover, "A fast quantum mechanical algorithm for database search", 28th Annual ACM Symposium on Theory of Computing, 1996, pp. 212–219, doi:10.1145/237814.237866

[26] M.-Z. Mina and E. Simion, "Threats to modern cryptography: Grover's algorithm", Preprints, September 2020, https://www.preprints.org/manuscript/202009.0677/v1

[27] M. Barger, M. Brohet, and F. Regazzoni, "Demonstrating Post-Quantum Remote Attestation for RISC-V Devices", 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), Valencia (Spain), March 2024, pp. 1–2, doi:10.23919/DATE58400.2024.10546557

[28] "Keylime", https://github.com/keylime/keylime

[29] Open Quantum Safe, "Liboqs", https://github.com/open-quantum-safe/liboqs