Experimental evaluation and expansion suggestion of Digital Signatures available on OP-TEE

Ewerton R. Andrade
Federal University of Rondonia (UNIR) and SIDIA Inst. of Science and Technology
Porto Velho / RO – Brazil
ewerton.andrade@sidia.com

Cristiano Coimbra Goes
SIDIA Inst. of Science and Technology
Manaus / AM – Brazil
cristiano.goes@sidia.com

João Bezerra Da R. Neto
SIDIA Inst. of Science and Technology
Manaus / AM – Brazil
joao.bezerra@sidia.com

César Augusto De S. Pastorini
SIDIA Inst. of Science and Technology
Manaus / AM – Brazil
cesar.pastorini@sidia.com

Janislley Oliveira De Sousa
SIDIA Inst. of Science and Technology
Manaus / AM – Brazil
janislley.sousa@sidia.com

***Abstract*—OP-TEE is an open-source project that uses security concepts and cryptography to create a Trusted Execution Environment (TEE) based on software isolation. Although being open-source and one of the main TEE for Android smartphones, it lacks an in-depth performance/security analysis of available algorithms, especially digital signatures. Therefore, in order to support next-generation of security applications, this work faces the challenge of improving the security of this trusted computing environment by (I) surveying and analyzing the security of the natively implemented digital signature algorithms on OP-TEE; (II) embedding and suggesting modern/secure alternatives as expansion; and (III) conducting experiments to check the performance of each these digital signature algorithms. As a result, users who are interested in adopting OP-TEE will be able to evaluate the security and functionality of various digital signature algorithms. Furthermore, our analysis confirms the necessity to include new algorithms, particularly post-quantum.**

***Index Terms*—OP-TEE, Digital Signatures, Analysis, Expansion, Experiments.**

I. Introduction

Trusted Execution Environments (TEEs) are designed as isolated and secure environments, operating alongside a device's primary operating system to ensure privacy, confidentiality, and security for applications and personal data [1]. Internally, devices equipped with TEEs handle two domains ("Normal World" and "Secure World") sharing the platform's resources.

These environments have been implemented in the most diverse kinds of computational devices, ranging from sensors with very low computational power [2] to servers and computers with high processing power [3]. Hence, it has become inevitable that smartphones implement Trusted Execution Environments, as since the second half of 2019 they have become the most used computational platform in the world [4]. With a specific emphasis on Android devices, which account for around 70% of this market [5]. Mobile devices' TEEs are used to provide a wide range of functionalities for their users, such as protecting sensitive data using encryption, fingerprint authentication, mobile payment, trusted user interface (TUI), Digital Rights Management (DRM), and lock mechanism (also referred to as SIM lock) [6].

Although the fact that there is an agreement on the minimal security requirements for a TEE, there are several differences between each of its implementations. Thus, these environments can be implemented using different resources, such as different methodological approaches, run on dedicated (co)processors, specific hardware, or even in isolated software environments.

It is important to note that several implementations of TEEs for Android smartphones have been developed, e.g.: Kinibi, by Trustonic [7]; QTEE, by Qualcomm [8]; Trusty, by Google [9]; Teegris, by Samsung [10] and OP-TEE, an open source project maintained by Linaro [11]. In this regard, it is worth noting that OP-TEE has become one of the most used execution environments in the market [12] because it is embedded in new devices based on the Armv8-A architecture and it is developed under the GNU General Public License (GPL).

Another important aspect to be commented on is that TEEs implement algorithms from several cryptographic classes, including symmetric and asymmetric ciphers, hash functions, message authentication codes, password hashing schemes, and digital signatures. Even though all classes of algorithms are significant, digital signatures play a key role in TEEs since all sensitive data is signed before it being moved between secure and normal domains, where critical operations are performed such as cryptography and secure payments. Furthermore, digital signatures are vital to end users' daily activities since they ensure the authenticity, integrity, and indisputability of the signed data [13].

The authors are grateful for the support offered by SIDIA R&D Institute in Mobile Cybersecurity project. This work was partially supported by Samsung, using resources of Informatics Law for Western Amazon (Federal Law No. 8.387/1991). Therefore, the present work disclosure is in accordance as foreseen in article No. 39 of number decree 10.521/2020.

Although TEE vendors provide specifications that describe their devices' algorithms and modes of operation [7], [8], [14], most of these documents do not provide a rigorous security analysis. But they all use cryptographic primitives and algorithms to ensure that only authorized applications can access the secure environment and perform privileged operations. Furthermore, although there are studies analyzing the correctness of these environments [15], performance analysis of cryptographic algorithms in these Trusted Execution Environments are scarce. Moreover, it is noteworthy that no initiatives were in progress discussing cryptographic alternatives for Android smartphone TEEs [16], although several modern and secure solutions raised in the scientific literature [17]–[19]. Particularly in the post-quantum area, which is one of the most prominent research fields today, due to the the development of a quantum algorithm with polynomial time to solve integer factorization and discrete logarithm [20], imminence of the quantum computer advent [21], and/or "steal now, decrypt later" method of operation [22].

However, software developers and smartphone vendors must be aware of the cryptographic algorithms available on the platform, as well as understand the performance impacts of their adoption. Thus, in order to cover these gaps in the study of digital signatures on OP-TEE (as open-source and one of the main TEE used on Android smartphones), this research aims to contribute by:

1) Surveying and analyzing the security of its natively implemented digital signature algorithms available on OP-TEE;
2) Embedding and suggesting modern and secure alternatives for future OP-TEE versions; and
3) Conducting experiments to check the performance of each these digital signature algorithms.

Thereby, the paper is organized as follows: In Section II, the materials and methods used for evaluating digital signature algorithms experiments are presented. The cryptography algorithms used in the OP-TEE is outlined in Section III. Section IV describes the alternative digital signatures employed in this study. The information gathered through experiments and the difficulties encountered in addressing them are presented in Section V. In Section VI, the performance metrics related to digital signatures are discussed. Finally, the conclusion is presented in Section VII.

II. Materials and Methods

To achieve the research objectives, the following software and tools were used: the source code editor Visual Studio Code; the C programming language for developing experimental scripts; the gcc and g++ compilers to generate execution codes; the QEMU emulator [23] to emulate OP-TEE on a personal computer; and to validate physical device with ARM TrustZone was used Raspberry Pi 3B Dev Kit [24].

The Raspberry Pi 3B embedded platform is equipped with Broadcom BCM2837 SoC (1GB RAM, quad-core ARM Cortex A53 running at 1.2GHz). The personal computer used in the experiments is equipped with the Linux Ubuntu 20.04 LTS 64 bits operating system, with 32GB of RAM, a Xeon E-2224G 3.5GHz processor, and 1TB of storage on an SSD. We utilized QEMU for programming and preliminary functionality testing [25], and we used Raspberry Pi for more in-depth studies on hardware. Emphasizing that these platforms are by the OP-TEE community [26] due to their versatility and fidelity to the results obtained on popular manufacturing devices.

Although the Raspberry Pi 3 processor has ARM TrustZone support, the use of OP-TEE or TrustZone capabilities within this board does not produce in a secure implementation. This happens because the Raspberry Pi 3B lacks secure memory and hence cannot protect memory [27]. Thus, given that DRAM is only protected against software, the access to it might be intercepted by an external digital analyzer.

Regardless of this physical platform constraint, the algorithms implementation for the several cryptographic classes are not compromised (e.g., TAs), since they are implemented in Secure World. Besides, it is important to conduct experiments to analyse the process of storing and managing device encryption keys, as well as associated time processing that might be employed on the operating system application. Additionally, the source code of the implemented TAs may be utilized on other TEE platforms.

Table I
Details about security levels and use recommendations

Label	Recommendation Level	Meaning	Security Strength
Dep	Deprecated	Should not be used. Specific care is needed for products already in the market.	$\leq$ 80 bits of security
Leg	Legacy	Should not be used for any new products/specifications. Products may already be in the market.	$\approx$ 112 bits of security
Rec	Recommended	Should be used for future (near/long-term) products/specifications.	$\geq$ 128 bits of security
PQC Rec	PQC Recommended	Suggested for post quantum use.	$\geq$ 128 bits of security in classical and quantum computers

Furthermore we use the research method called exploratory study based on experiments [28], because this research aims to prospect possibilities and scenarios that have not yet been explored in TEEs. To do this, we perform experiments with several cryptographic algorithms to analyze different metrics.

In addition, due to the characteristics and small amount of data, a qualitative approach was used [28], since this research was focused on obtaining a broad understanding and the performance impact of the cryptographic algorithms natively implemented on OP-TEE. To obtain a realistic and independent view of the security of these algorithms, we use a bibliographic review method based on articles and institutional reports from regulatory agencies to analyze the security of the algorithms [28].

To facilitate the readability and interpretation of our security analyses, the algorithms were grouped into security levels and use recommendations, namely: Deprecated, Legacy, Recommended, and Post-Quantum Recommended. In this sense, Table I shows details of each level.

Post-Quantum Cryptography is the area of study where cryptographic systems based on intractable problems in classical and quantum computers are researched and developed [29]. This is due to the fact that, in 1997, mathematician Peter Shor created a polynomial time algorithm (i.e., with an execution time of less than 80 bits of security) to solve discrete logarithm and integer factorization problems using a quantum computer [20]. It is worth remembering that integer factorization and discrete logarithm are the basis of the main cryptographic systems used today (e.g., RSA, DSA, ElGamal, ECDSA, EdDSA, etc.) [29].

III. Survey and analysis of the Digital Signatures implemented on OP-TEE

OP-TEE is a software-based Trusted Execution Environment, which means it shares the same hardware but has isolated software components that interact using APIs and secure system calls. More specifically, each of the CPU's

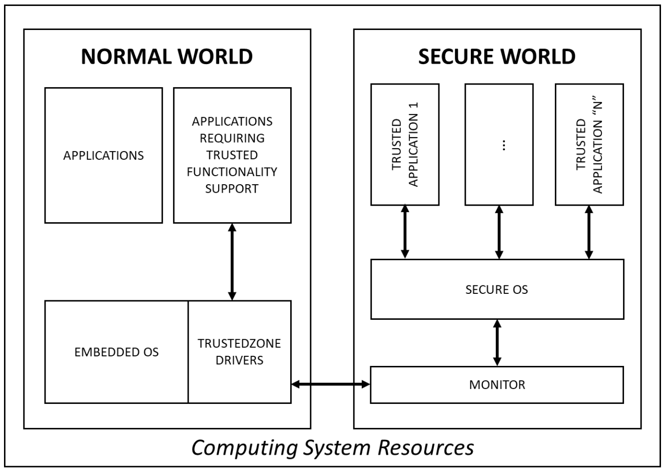

Figure 1. OP-TEE architecture [11]

physical cores provides two virtual cores, one considered non-secure and the other secure, each of them is used for the "Normal World" and the "Secure World", respectively. In the "Normal World" of the processor (also called Rich Execution Environments), conventional applications and the operating system built by the manufacturer are executed (such as Android and commercial apps). This conventional applications can be also called as Client Applications (CAs).

While the "Secure World" runs the Trusted Applications (TAs) and manages communication between the two worlds. The secure area also runs the secure operating system itself, OP-TEE, which implements how applications running on it can generate, operate and persist data cryptographically securely.

Additionally, OP-TEE has a hardware-level separation between the "Normal World" and the "Secure World" within the same processor: TrustZone. In this way, OP-TEE protects security applications and privileged functions that can be called by Client Applications (CAs). An OP-TEE diagram can be seen in Figure 1.

One of TrustZone's most used properties is access to hardware-based cryptographic key generation. Each CPU used in the device with TrustZone has a unique key written into the hardware during the manufacturing. This hardware key is accessible only to the encryption driver that runs with privileges in the TrustZone. As a result, these keys are accessible to any application in the "Secure World", and only applications in the "Secure World" can derive secondary keys based on the hardware's unique key, which can be used to protect resources (e.g., files) so that they cannot be opened in the "Normal World". Thus, keys are bound to hardware and it is not possible to decrypt files or any resources on another device.

As previously mentioned on Section I, digital signatures are vital to end users' daily activities that need authenticity, integrity, and indisputability. Furthermore, all sensitive data transferred through TEEs' APIs are digitally signed. In other words, the data accessed by applications that operate between the standard operating system of the "Normal World" and the secure applications (TAs) of the "Secure World" are digitally signed. Consequently, these interfaces are the major security risk for trusted operating systems, since they are the only entries to the TEE. Thus, strengthening your security is the main way to ensure that accesses are not vulnerable to attacks.

In this sense, survey and analysis are essential to understand the trade-offs and keep the platform secure. Therefore, using documentation released by the community that maintains the project, it was verified that OP-TEE natively implements the digital signature described in Table II.

Analyzing this Table, we can verify that OP-TEE natively implements only one digital signature for legacy applications (DSA with 2048-bit keys) and several other algorithms recommended for current applications. In addition, it does not provide any deprecated digital signature algorithm. However, by implementing only cryptographic algorithms based on integer factorization and discrete logarithm, all of them will be broken in the eventual advent of the quantum computer.

Table II
DIGITAL SIGNATURE ALGORITHMS NATIVELY IMPLEMENTED ON OP-TEE AND THEIR SECURITY LEVELS (ADAPTED FROM [14])

Algorithm	Security Strength (Table I for details)			
	Dep	Leg	Rec	PQC Rec
DSA 2048		✓		All these algorithms will be broken and will need to be replaced
DSA 3072			✓	
EdDSA Curve25519			✓	
ECDSA P-256			✓	
ECDSA P-521			✓	

IV. ALTERNATIVE DIGITAL SIGNATURES

Due to the security requirements, the development of a new cryptographic algorithm can be an extremely challenging task [30]. First, cryptography is a highly specialized field, which requires deep knowledge in mathematics, computation theory, number theory, graph theory, computer networks, and several other related areas.

Furthermore, security algorithms are constantly evolving and susceptible to new vulnerabilities and regularly discovered attacks. This implies the fact that an algorithm that is considered secure today can be broken tomorrow. Another obstacle is related to costs. As the research and development of a secure alternative can take years, the cryptographic algorithm creation can be expensive and time-consuming. Not forgetting to mention that tests and experiments can require significant resources.

The standardization barrier is the last one, as it serves little purpose to develop a novel algorithm if it is not adopted by other individuals and organizations. Cryptographic algorithms must pass strict performance and security testing in order to be standardized. This undoubtedly entails the assessment of a number of experts who were not involved in its formulation. In order for it to be embraced widely, it must first go through an international standardization procedure. Thankfully, there are various methods to make this process easier, with the major one being the usage of open-source algorithms that have taken part in cryptography competitions [30].

In this sense, using cryptographic competitions to find new secure and efficient digital signatures for Android smartphones, it is possible to use the recently selected algorithms in the Post-Quantum NIST competition [18]: Falcon, Dilithium, and Sphincs+ [31]–[33]. These digital signatures are secure alternatives to current public-key cryptographic standards [18]. In other words, modern and safe algorithms that can be incorporated into TEEs, for example the OP-TEE, which is one of the research objectives of this work.

V. EXPERIMENTS

Using the collected data and previously described methods as a basis, a strategy was developed to use the OP-TEE architecture to verify (I) the performance of each of the natively implemented digital signature algorithms (i.e., DSA, EdDSA, ECDSA); (II) as well as the viability of embedding and performance of the post-quantum digital signatures Falcon, Dilithium, and Sphincs+.

More specifically, we use the reference source codes provided by the authors to embed Falcon, Dilithium, and Sphincs+ algorithms [31]–[33]. Emphasizing that these codes were utilized and completely evaluated throughout the NIST Post-Quantum Cryptography Standardization Process [18], and do not employ any special instructions set (e.g., NEON) to improve the runtime.

Besides, we use the manuals available by the OP-TEE community as a development basis to create a benchmark application that runs in the Normal World and call the resources of a trusted benchmark application running in the Secure World [11], [26]. Emphasizing that all tested algorithms (i.e., DSA, EdDSA, ECDSA, Falcon, Dilithium, and Sphincs+) are embedded and executed in these new trusted benchmark applications that run in the Secure World, being created a TA for each digital signature and its configuration. Therefore, an overview of this strategy designed and implemented in OP-TEE can be represented as shown in Figure 2.

As discussed before on Section II, we utilized QEMU for programming and preliminary functionality testing emulated, and we used Raspberry Pi for more in-depth studies on actual physical hardware. Additionally, there are performance differences between the Secure World and Normal World processor modes while running applications since each processor modes run distinct operating systems that might change the system performance conditions in accordance with their own standards. In this context, a setup that comes as close to the real-world environment as possible should be used to compare how effectively the algorithms operate in devices like smartphones. Therefore, the closest open-source alternative was chosen for this study, which is OP-TEE running alongside a Linux System on QEMU and Raspberry Pi.

It is important to mention that this research employed clock time to check how the performance of OP-TEE varied on

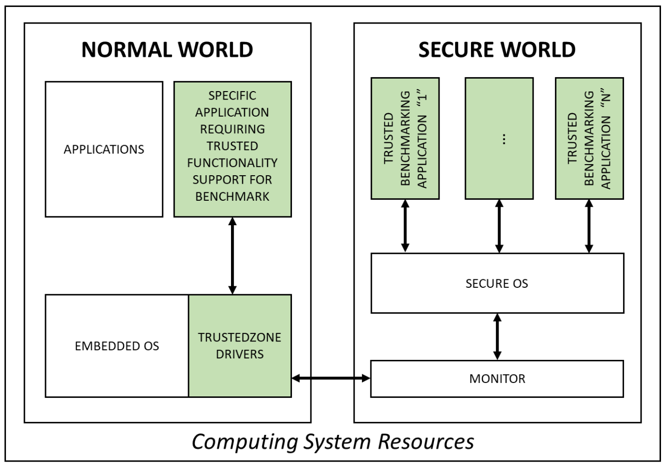
NORMAL WORLD
APPLICATIONS
SPECIFIC APPLICATION REQUIRING TRUSTED FUNCTIONALITY SUPPORT FOR BENCHMARK
EMBEDDED OS
TRUSTEDZONE DRIVERS
SECURE WORLD
TRUSTED BENCHMARKING APPLICATION "1"
...
TRUSTED BENCHMARKING APPLICATION "N"
SECURE OS
MONITOR
Computing System Resources

Figure 2. OP-TEE architecture adapted to our experiments

different platforms (QEMU and Raspberry Pi). As a result, we wanted to better understand how long it takes the algorithms to perform in real time. Remembering that this metric is the same used during NIST Post-Quantum Cryptography Standardization Process [34].

Another important factor to consider is the time added by communication via Normal World and Secure World (overhead). Because it is essential to understand the cost in execution time associated with using TEEs to protect the process. Noting that this overhead is purely related to communication delay and that the application executing in the Normal World does not execute any cryptographic operations (i.e., just call the trusted benchmark application on Secure World).

As for the requirements for running digital signature experiments, we have to keep in mind that these methods of data authentication are commonly compared with the physical signature process on paper. In essence, the objectives are similar, i.e., both physical and digital process aims for the authentication of the sender, the non-repudiation and the integrity of a legitimate signature [13]. Regularly, the digital signature algorithm can be formalized as Definition 1.

Definition 1 (Digital Signature Scheme)**.** Is a scheme defined as follows [35]:

- **The key-generation algorithm**: is a probabilistic algorithm which given a system parameter outputs a pair of matching public and private keys $(PubKey, PriKey)$.
- **The signing algorithm**: is a probabilistic algorithm which takes the message M to be signed and a private key $PriKey$, and returns a signature $\sigma = Sig_{PriKey}(M)$.
- **The verification algorithm**: is a algorithm which takes a message M, a candidate signature σ and $PubKey$, and returns a bit obtained through $Ver_{PubKey}(M, \sigma)$. The signature is accepted only if the bit is equal one. Otherwise, it is rejected.

Thus, to measure the amount of resources used by these algorithms and consequently understand the impacts of their adoption, it is essential to measure the time spent by each of these steps and verify the amount of memory required for their execution. In this way, the following metrics must be collected:

- **Execution time** ***(on Secure World and Normal World)***
  - Time to generate a key pair;
  - Time to sign a message using the private key;
  - Time to verify a signed message using the public key;
  - Time overhead by communication between isolated areas.
- **Memory consumption**
  - Length of the private key;
  - Length of the public key;
  - Length of the signature.

Furthermore, the experiments must be performed until the standard deviation reaches an acceptable range to avoid inconsistencies between the collected data. Thus, for this work, it was defined that all algorithms must be executed at least 20 times and have a standard deviation of less than 5% (five percent) in the collected samples. Additionally, it must run as many times as necessary until this standard deviation is reached.

VI. Results

Following what was established, we embedded and performed experiments with the digital signatures DSA, EdDSA, ECDSA, Falcon, Dilithium, and Sphincs+; and the results of these experiments are summarized in Tables III, IV and V.

For clarity, each algorithm's properties and parameters are listed next to it, so that everything is clearly explained. For example, the number that follows DSA, is specifying the size of the prime numbers used. In the case of EdDSA and ECDSA, are described the underlying elliptical curves used. While in Falcon, the numbers that follow it refer to the size of the internal blocks used. On the other hand, in Dilithium, the authors chose to make the standardized security levels (the higher the value, the more secure). Lastly, in the case of Sphincs+, the information that follows describes the different settings supported by the algorithm.

Besides that, it is also important to highlight that Sphincs+ supports different settings and underlying hash functions [33]. However, we decided to present only the best results data for readability and objectivity. Therefore, the Sphincs+ experiments were performed using only the underlying hash function that presents the best results, in this case, SHA-256 [33].

In this sense, results shown in Table III and Table IV, we can make several observations about the algorithms tested in the QEMU and Raspberry Pi environments:

- The DSA algorithm, with key sizes of 2048 and 3072, shows a similar pattern in both environments. The time taken to generate a key and perform verification is higher than the time taken to generate a signature. As the key size increases, the time for each operation also increases.
- The EdDSA algorithm with the Curve25519 curve shows very little time difference between operations in both environments, indicating that it's highly efficient. Its execution time is much less compared to other algorithms.
- The ECDSA algorithm, with the P-256 and P-521 curves, has a higher execution time in comparison to EdDSA, with the time increasing significantly with the curve size. Similar to DSA, the time taken for key generation and verification is longer than the signature generation time.
- The Falcon algorithm, with internal block sizes of 512 and 1024, shows very high execution times for key generation and signature generation, but an extremely low time for verification. The Falcon-1024 variant requires significantly more time than Falcon-512 for all operations.
- The Dilithium algorithm at both Level 3 and Level 5 security levels performs efficiently, with relatively small execution times for all operations. The times between the two security levels do not vary greatly.

Table III
OVERVIEW OF EVALUATED DIGITAL SIGNATURES INCLUDING PERFORMANCE BENCHMARKS ON QEMU

Algorithm	*Execution time on QEMU*[*]								
	Normal World			*Secure World*			*Time Overhead*		
	Key-gen	**Signature**	**Verification**	**Key-gen**	**Signature**	**Verification**	**Key-gen**	**Signature**	**Verification**
DSA 2048	197.0	708.4	1470.7	195.1	704.1	1466.5	1.9	4.3	4.2
DSA 3072	410.0	756.0	1406.2	408.3	752.5	1402.4	1.7	3.5	3.8
EdDSA Curve25519	31.8	34.5	57.0	30.0	34.2	53.3	1.8	4.3	3.7
ECDSA P-256	909.4	988.9	664.2	908.0	985.6	659.1	1.4	3.8	5.1
ECDSA P-521	3797.3	4200.6	2821.2	3796.8	4197.1	2817.1	0.5	3.5	4.1
Falcon-512	4285.7	4039.2	29.0	4281.5	4035.0	24.0	4.2	4.2	5.0
Falcon-1024	15999.0	8223.0	54.9	15997.3	8220.1	50.5	1.7	2.9	4.4
Dilithium – Level 3	16.7	30.5	15.1	14.1	28.0	13.0	2.6	2.5	2.1
Dilithium – Level 5	19.7	28.1	21.2	17.4	25.8	19.4	2.3	2.3	1.8
Sphincs+-128f simple	39.6	870.5	48.8	37.1	867.1	46.0	2.5	3.4	2.8
Sphincs+-192f simple	63.6	2087.1	88.0	60.0	2084.0	85.0	3.6	3.1	3.0

[*]In milliseconds.

Table IV
OVERVIEW OF EVALUATED DIGITAL SIGNATURES INCLUDING PERFORMANCE BENCHMARKS ON RASPBERRY PI

Algorithm	*Execution time on Raspberry Pi*[*]								
	Normal World			*Secure World*			*Time Overhead*		
	Key-gen	**Signature**	**Verification**	**Key-gen**	**Signature**	**Verification**	**Key-gen**	**Signature**	**Verification**
DSA 2048	41.8	98.9	170.0	31.2	87.2	157.3	10.6	11.7	12.3
DSA 3072	63.0	100.2	175.2	52.4	88.6	163.1	10.6	11.6	12.1
EdDSA Curve25519	66.4	69.5	113.0	55.0	57.2	100.5	11.4	12.3	12.5
ECDSA P-256	481.8	488.7	340.3	471.6	477.4	326.0	10.2	11.3	14.3
ECDSA P-521	2064.6	2076.0	1245.1	2056.3	2065.9	1234.0	8.3	10.1	11.1
Falcon-512	1299.0	927.0	77.8	1248.2	875.4	24.0	50.8	51.6	53.8
Falcon-1024	3845.2	1930.9	80.6	3797.6	1884.0	31.2	47.6	46.9	49.4
Dilithium – Level 3	16.6	26.0	27.5	3.3	12.1	13.0	13.3	13.9	14.5
Dilithium – Level 5	18.0	32.9	30.2	4.4	18.0	15.7	13.6	13.9	14.5
Sphincs+-128f simple	60.0	557.2	59.3	32.0	530.2	31.7	28.0	27.0	27.6
Sphincs+-192f simple	71.3	1003.6	73.9	43.2	977.0	47.8	27.9	26.6	26.1

[*]In milliseconds.

- The Sphincs+ algorithm, in both the 128f simple and 192f simple settings, shows a high execution time for signature generation compared to key generation and verification. As the security setting increases, the time for all operations also increases.
- The time overhead on Raspberry Pi was significantly higher than on QEMU.
- Although not negligible in terms of time, the overhead does not appear to make the use of digital signatures in TEEs prohibitive.

Excepting EdDSA Curve25519, on comparing the two platforms, all algorithms take a significantly short time in the Raspberry Pi than in the QEMU environment. This shows that these algorithms might perform better on the dedicated physical hardware compared to the emulated environment. EdDSA and Dilithium consistently perform well in both platforms, with reasonably short execution times across all operations. On the other hand, algorithms such as Falcon and ECDSA (P-521) show long execution times, especially in QEMU.

However, as previously discussed, it is not enough only measure its execution time if the goal is to measure the impact of specific algorithm adoption. Thus, we also analyzed memory consumption, summarized in Table V.

Hence, from this table, it is possible to realize that:

- Among all analyzed algorithms, Falcon presented the highest memory consumption for storing the private key.
- Among all analyzed algorithms, Dilithium presented the highest memory consumption for storing the public key.
- Among all analyzed algorithms, Sphincs+ presented the highest memory consumption for storing the signatures.
- Among all the analyzed algorithms, the algorithms based on elliptical curves (EdDSA and ECDSA) presented the lowest average memory consumption to store the necessary structures for the operation of their algorithms (keys and signature).
- Following the trend of several studies [13], [18], [29], the post-quantum algorithms (Falcon, Dilithium, and Sphincs+) presented the highest average memory con-

Table V
MEMORY CONSUMPTION OF THE EVALUATED DIGITAL SIGNATURES

Algorithm	*Memory consumption***		
	Private Key	Public Key	Signature
DSA 2048	2048	2048	2048
DSA 3072	3072	3072	3072
EdDSA Curve25519	256	256	512
ECDSA P-256	256	256	512
ECDSA P-521	521	521	1042
Falcon-512	60424	7176	5328
Falcon-1024	111624	14344	10240
Dilithium – Level 3	32000	15616	26344
Dilithium – Level 5	38912	20736	36760
Sphincs+-128f simple	512	256	136704
Sphincs+-192f simple	768	384	285312

**In bits.

sumption to store the necessary structures for the operation of their algorithms (keys and signature).
- Although having the highest memory consumption for storing the signatures, Sphincs+ is comparable to algorithms based on elliptical curves (EdDSA and ECDSA) for storing the generated keys.

In addition, crossing the data presented in Tables III, IV and V, it is possible to infer that:
- When comparing the conventional algorithms DSA and ECDSA to modern alternatives like EdDSA Curve25519, Falcon, Dilithium, and Sphincs+; they present promising results in both QEMU and Raspberry Pi environments.
- In both environments, post-quantum algorithms such as Falcon and Dilithium performed particularly well in terms of key generation and verification times, displaying competitive results against traditional algorithms.
- While some post-quantum algorithms (Falcon, for instance) demonstrated a relatively high key generation time, they compensated for it with faster signature verification execution times.
- The key generation procedure does not run frequently since the users commonly use the same key pair to sign several messages.
- When comparing the Normal and Secure worlds in both QEMU and Raspberry Pi, the time overhead is minimal, suggesting that the trusted environment execution does not dramatically affect the algorithms performance.
- In both environments, Dilithium Level 3 and 5 showed promising results, especially considering that the higher security level (Level 5) only caused a slight increase in execution times.
- The algorithms present better execution time in the Raspberry Pi environment compared to QEMU, indicating that physical platforms may offer enhanced performance for these algorithms.
- Optimizations for Single Instruction Multiple Data – SIMD (e.g., NEON) can reduce the execution time of post-quantum algorithms even more.
- The post-quantum algorithms experiments reinforce the trend of good execution time but higher memory consumption.

The time overhead associated with TEE execution includes setup and teardown of the TEE session. In OP-TEE, the overhead is primarily due to the communication channel between Secure World and Normal World. Due to the introduction of two time-consuming mechanisms by this communication channel, the invoked functions are slowed down. The connection maintenance of the Secure World (e.g., initializing and finalizing context, opening and closing sessions) and the execution of partitioned functions in the Secure World (e.g., allocating and releasing shared memory, algorithm cryptographic operations) slow the invoked functions down.

These results underline the viability of embedding post-quantum cryptographic algorithms in future versions of OP-TEE, suggesting that the minor increase in execution time is outweighed by the increased security these algorithms provide, particularly in light of the advent of quantum computing.

Additionally, it is important to note that the results of our experiments may not be applicable to other Android TEEs (such as Kinibi, QTEE, Trusty, Teegris, etc.). Because the technological aspects of each environment have an effect on performance. Moreover, we must keep in mind that embedding a software application in the Secure World of a device with OP-TEE capability is not the same as running it in the Normal World. Due to the execution time in the Real World might differ depending on the operating system, programming language, and other factors.

Consequently, it is reasonable to draw the conclusion that no single digital signature method is the best in every situation. Thus, each developer and smartphone vendor must consider the security requirements of their platforms, trade-offing their advantages and limitations. It is also clear that each developer must consider the overhead added in the execution time of an application that uses resources from the Secure World, verifying whether this makes its use unfeasible.

VII. Conclusion and Future Work

This research gives an analysis aimed at supporting software developers and smartphone vendors in understanding the landscape of digital signatures that may be utilized in the TEEs of their platforms. To do this, we (I) surveyed and analyzed the security of the natively implemented digital signature algorithms on OP-TEE, (II) embedded and suggested modern and secure alternatives for future OP-TEE versions, and (III) conducted experiments to check the performance of each of these algorithms.

With the aid of the surveyed information, it was feasible to confirm that OP-TEE only natively implements digital signatures based on discrete logarithms and integer factorization. Which one of them is already depreciated and the others will eventually break with the eventual advent of the quantum computer.

By examining the results of our experiments, it is feasible to verify that traditional cryptographic algorithms like DSA and ECDSA are being outperformed by modern alternatives such as EdDSA Curve25519, Falcon, Dilithium, and Sphincs+. This was particularly true in the Raspberry Pi board, where the algorithms performed significantly better than in the QEMU emulator. Notably, even post-quantum algorithms such as Falcon and Dilithium showed relatively slower key generation times and compensated for this by offering considerably faster signature verification times.

Furthermore, even at higher security levels, post-quantum algorithms like Dilithium showed promising results. These findings highlight the potential of incorporating these advanced post-quantum cryptography algorithms into future versions of OP-TEE. It is because (I) these algorithms are in the process of standardization [18]; (II) the experiments show that they have good performance in terms of execution time; (III) memory is not a critical point in current smartphones; (IV) the advent of the quantum computer is getting closer [21]; (V) there is a large number of users concerned with maintaining the secrecy of their information over the next few years, especially in the business world.

Additionally, the results imply that a modest increase in execution time is a minimal price to pay for the increased security provided by post-quantum cryptographic algorithms. And their incorporation into the OP-TEE platform has the potential to significantly contribute to the development of future secure and dependable cryptographic systems [36].

We also see possibilities for further researches given the wide range of cryptographic classes natively provided on OP-TEE, including symmetric ciphers, asymmetric ciphers, hash functions, message authenticator codes, password hashing schemes, and more. This might include running further experiments with alternative algorithms, suggesting improvements to already in use algorithms (e.g., employing SIMD instructions or hardware methods to optimize), or introducing new algorithms into the platform.

References

[1] O. Hosam and F. BinYuan, "A Comprehensive Analysis of Trusted Execution Environments," in 2022 8th ITT. IEEE, 2022, pp. 61–66.

[2] D. Oliveira, T. Gomes, and S. Pinto, "uTango: an open-source TEE for IoT devices," IEEE Access, vol. 10, pp. 23 913–23 930, 2022.

[3] K. Chen, "Confidential High-Performance Computing in the Public Cloud," IEEE Internet Computing, vol. 27, no. 1, pp. 24–32, 2023.

[4] STATCOUNTER, "Desktop vs Mobile vs Tablet Market Share Worldwide," Available at: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201501-202304, 2023.

[5] ——, "Mobile Operating System Market Share Worldwide," Available at: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201001-202303, 2023.

[6] C. Shepherd, K. Markantonakis, N. van Heijningen, D. Aboulkassimi, C. Gaine, T. Heckmann, and D. Naccache, "Physical fault injection and side-channel attacks on mobile devices: A comprehensive analysis," Computers & Security, vol. 111, p. 102471, 2021.

[7] TRUSTONIC, "Kinibi v510A," https://www.commoncriteriaportal.org/files/epfiles/Trustonic-Kinibi-510A-ST-1.2.pdf, 2022.

[8] QUALCOMM, "Guard Your Data with the Qualcomm® Snapdragon™ Mobile Platform," Available at: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/guard_your_data_with_the_qualcomm_snapdragon_mobile_platform2.pdf, 2019.

[9] Google, "Trusty tee," Available at: https://source.android.com/docs/security/features/trusty, 2023, accessed: 2023-07-31.

[10] Samsung, "Samsung teegris," Available at: https://developer.samsung.com/teegris/overview.html, 2023, accessed: 2023-07-31.

[11] Linaro, "OP-TEE," Available at: https://op-tee.org/, 2023.

[12] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," ACM computing surveys (CSUR), vol. 51, no. 6, 2019.

[13] E. R. Andrade, "Proposta de aprimoramento para o protocolo de assinatura digital Quartz," Master's thesis, Computer Science, Instituto de Matemática e Estatística, Universidade de São Paulo, 2013.

[14] Global Platform, "Cryptography Recommendations for TEE Internal Mechanisms & TEE Internal Core API Specification v1.3.1," https://globalplatform.org/specs-library/?filter-committee=tee, 2023.

[15] NIST, "Qualcomm® Trusted Execution Environment (TEE) Software Cryptographic Library. FIPS 140-2 Non-Proprietary Security Policy. Version: 1.1. Date: 2021-10-27," 2021.

[16] ECRYPT, "eBACS: ECRYPT Benchmarking of Cryptographic Systems," Available at: https://bench.cr.yp.to/, 2019, accessed: 2023-07-31.

[17] CAESAR, "CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness," Available at: https://competitions.cr.yp.to/caesar.html, 2019, accessed: 2023-07-31.

[18] NIST, "Post-Quantum Cryptography (CSRC)," Available at: https://csrc.nist.gov/projects/post-quantum-cryptography, 2023, ac.: 2023-07-31.

[19] PHC, "Password Hashing Competition," https://www.password-hashing.net/, 2019, accessed: 2023-07-31.

[20] 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, 1997.

[21] ScienceDaily, "Your source for the latest news – Quantum Computers," https://www.sciencedaily.com/news/computers_math/quantum_computers/, 2023.

[22] IDC, "IDC TechBrief: Applying Post-Quantum Cryptography to Data Protection to Enhance Digital Trust," Available at: https://www.idc.com/getdoc.jsp?containerId=US50789923, 2023, accessed: 2023-07-21.

[23] F. Bellard, "Qemu, a fast and portable dynamic translator." in USENIX annual technical conference, FREENIX Track, vol. 41. USA, 2005.

[24] OP-TEE documentation, "Raspberry pi 3," https://optee.readthedocs.io/en/latest/building/devices/rpi3.html, 2023, accessed: 2023-07-31.

[25] QEMU, "QEMU: A generic and open source machine emulator and virtualizer," https://www.qemu.org/, 2023, accessed: 2023-07-31.

[26] KE, "Kickstart Embedded: OP-TEE: Part 4 – Writing Your First Trusted Application," https://kickstartembedded.com/, 2022, ac.: 2023-07-31.

[27] R. P. Forum, "Why is op-tee for raspberry pi3 insecure?" Available at: https://optee.readthedocs.io/en/latest/building/devices/rpi3.html, 2023, accessed: 2023-06-31.

[28] E. Khan and H. Anwar, Research Methods of Computer Science. Laxmi Publications Pvt. Limited, 2015.

[29] D. Bernstein, J. Buchmann, and E. Dahmen, Post-Quantum Cryptography. Springer Berlin Heidelberg, 2009.

[30] N. Ferguson, B. Schneier, and T. Kohno, Cryptography engineering: design principles and practical applications. John Wiley & Sons, 2011.

[31] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-Dilithium home page," Available at: https://pq-crystals.org/dilithium/index.shtml, 2021, accessed: 2023-07-31.

[32] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Fast-Fourier Lattice-Based Compact Signatures over NTRU," https://falcon-sign.info/, 2017, accessed: 2023-07-31.

[33] A. Hülsing, J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. Westerbaan, "Sphincs+: Statless hash-based signatures – home page," Available at: https://sphincs.org/, 2022, accessed: 2023-07-31.

[34] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, C. Miller, D. Moody, R. Peralta et al., "Status report on the third round of the nist post-quantum cryptography standardization process," US Department of Commerce, NIST, 2022.

[35] K. Sakumoto, T. Shirai, and H. Hiwatari, "On provable security of UOV and HFE signature schemes against chosen-message attack," in PQCrypto 2011, Taiwan. Springer, 2011, pp. 68–82.

[36] Olivier Van Nieuwenhuyze, "Moving forward with confidence: preparing for a phased migration to Post-Quantum Cryptography," https://globalplatform.org/, 2022.