Received May 20, 2021, accepted June 25, 2021, date of publication July 27, 2021, date of current version August 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3100551

O2MD^2: A New Post-Quantum Cryptosystem With One-to-Many Distributed Key Management Based on Prime Modulo Double Encapsulation

RICARDO NEFTALI PONTAZA RODAS[1], YING-DAR LIN[2], (Fellow, IEEE), SHIH-LIEN LU[3], AND KEH-JENG CHANG[4]
[1]College of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan
[2]Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan
[3]PieceMakers Technology, Inc., Hsinchu 30070, Taiwan
[4]Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu 30075, Taiwan

Corresponding author: Ricardo Neftali Pontaza Rodas (pontaza.ricardo.05g@g2.nctu.edu.tw)

This work was supported in part by Taiwan Semiconductor Manufacturing Company (TSMC), and in part by the Ministry of Science and Technology (MOST), Taiwan.

ABSTRACT Polynomial-time attacks designed to run on quantum computers and capable of breaking RSA and AES are already known. It is imperative to develop quantum-resistant algorithms before quantum computers become available. Computationally hard problems defined on lattices have been proposed as the fundamental security bases for a new type of cryptography. The National Institute of Standards and Technology (NIST) recently hosted the *Post-Quantum Cryptography Standardization* project, aiming to create a roster of innovative post-quantum cryptosystems. These candidates have been publicly available for testing since early 2017. As they are currently under analysis, new proposals are still desirable. As such, we use the ring learning with errors (RLWE) problem combined with arithmetic functions to propose the O2MD^2 cryptosystem, which provides a one-to-many private/public key architecture having a distributed key refresh for a network of users while working on multiple polynomial rings over different prime order fields. Our solution has three different frameworks that reach AES-256 equivalent security, and provides message integrity and message authenticity verifications. We compare our solution's speed against the speed of the twenty-six different implementations from seven popular candidates in the NIST project, and our cryptosystem performs from 2 to 4 orders of magnitude faster than them. We also propose six different implementations that reach the security levels 1, 3 and 5 proposed in the NIST competition. Finally, we used the NIST *Statistical Test Suite* to verify the indistinguishability of our produced ciphertexts against randomly generated noise.

INDEX TERMS Abstract algebra, cryptographic protocols, quantum cryptography, lattices, O2MD^2, post-quantum.

I. INTRODUCTION

Currently cryptographic protocols rely heavily on algorithms that are expected to be broken when first generation quantum computers become a reality. Algorithms which depend on computationally hard problems like the integer factorization problem (such as RSA) and the elliptic curve discrete logarithm problem (for elliptic curves cryptography) will be seriously affected. Shor's algorithm can be used for these attacks, as it is a fast quantum algorithm that can find the prime factorization of any positive integer N [1]. Variations of Shor's algorithm lead to equally significant results. Beauregard's variation uses $2n+3$ qubits to perform $O(n^3 \log n)$ operations to factorize $N = pq$ with p and q primes, and it can also be modified to perform $n^{2+o(1)}$ if N fits into n bits. Also, by a simple algebraic transformation, Shor's algorithm can be used to solve the discrete logarithm problem [2].

Besides Shor's algorithm, Grover's algorithm has also been studied. For a black box function f and a known output value $y = f(x)$, this algorithm finds the input x that generates that specific output y by using $O(\sqrt{N})$ evaluations, where N is the total number of possible inputs [3]. It can also be used to find the roots of a function in a finite set, when we can assume that one out of every N inputs is a root. This can be used to attack AES implementations. For example, when encoding two public messages m_1 and m_2 under a secret 128-bit AES key k, we can produce a 256-bit ciphertext $c = (\text{AES}_k(m_1), \text{AES}_k(m_2))$. Then by defining

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Verticale.

$f(x) = (\mathrm{AES}_x(m_1), \mathrm{AES}_x(m_2)) - c$, Grover's algorithm can be used to find a root of f using about 2^{64} quantum evaluations [4]. Grover's algorithm can also be used to attack cryptosystems that have a 2^{128} bits security, such as the 128-bit AES. Nowadays it is considered that doubling the size of the keys is sufficient to prevent quantum attacks, but it is unknown if it is a reliable long-term solution.

A. REPLACING THE INTEGER FACTORIZATION PROBLEM

Because Shor's and Grover's algorithms-based attacks will have a speed advantage when attacking nowadays secure instances of RSA and AES, different solutions have been proposed. The first one is to double the size of the private keys of existing cryptosystems. Some authors speculate that doubling the size of current secure instances of RSA and AES should be enough to resist attacks by quantum computers, but if the development of quantum-based hardware has a behavior similar to Moore's law, doubling the size of the private keys will prove to be only a temporary solution [2].

Cryptosystems based on computationally hard lattice problems have been proposed as new candidates to replace the traditional algorithms based on the integer factorization and discrete logarithm problems [5], [6]. Lattice-based cryptography provides a wide diversity of algebraic structures that can lead to extremely efficient and secure cryptosystems, making it well received by the cryptographic community [7], [8]. One of the major contributions of this new field of research is the learning with errors (LWE) problem which nowadays has evolved into different variations, such as the module learning with errors (MLWE) problem, the integer module learning with errors (I-MLWE) problem, and the ring learning with errors (RLWE) problem. All these problems have been used as the fundamental security bases for different cryptosystems [9]– [14]. In this work, we focus on the ring learning with errors (RLWE) problem, which is defined over a polynomial ring, and asks to identify polynomials of a specific form from a list of polynomials containing both constructed and randomly generated ones. This problem has been widely studied, and polynomial rings are ideal to build robust cryptosystems.

B. O2MD^2: A ONE-TO-MANY PRIVATE/PUBLIC KEY QUANTUM-RESISTANT CRYPTOSYSTEM WITH DISTRIBUTED KEY REFRESH

In this paper we utilize the ring learning with errors (RLWE) problem to design a new quantum-resistant public key cryptosystem which we refer to as O2MD^2. O2MD^2 stands for its capabilities: A One-to-Many private/public key architecture (referred as O2M), a Distributed key refresh capability for all the users, and the use of different polynomial quotient rings over fields of prime order, which we refer as Double encapsulation using prime modulo operations. The O2MD^2 cryptosystem is based on arithmetic functions and some custom algebraic structures that we call p-bases, and uses the quotient rings $\mathbb{Z}_{b+1}[x]/\langle x^m - 1\rangle$, $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$ where m, b, p_1, p_2 are positive integers and p_1, p_2 primes.

We use ordered sets of positive integers and one-way hash functions that we define as **sessions** in order to generate the private and public keys. Our private keys' coefficients are non-negative integers and their values are not as strictly bounded as the coefficients in other post-quantum cryptosystems. This characteristic provides security against attacks that focus on guessing the private key while using a given session. Also, the use of our private function f allows us to generate multiple persistent keys for different sessions. Our encryption and decryption use operations in different polynomial quotient rings, and they are performed in polynomial time. The produced ciphertexts will be generated in a way that they are indistinguishable against random noise, characteristic that is explored by the use of the NIST *Statistical Test Suite* later in this document.

The O2MD^2 cryptosystem consists of three frameworks: **O2MD2-I**, **O2MD2-II** and **O2MD2-III**, where each framework provides stronger security capabilities than its predecessor. The **O2MD2-I** framework is reliable, secure, and easy to implement. The **O2MD2-II** framework is based on its predecessor, and provides message integrity verifications. The **O2MD2-III** framework is the strongest framework out of the three versions, and provides simultaneously both message integrity and message authenticity verifications.

For traditional public key cryptosystems, one private key k_{priv} generates one public key k_{pub} and that public key k_{pub} can be only generated from the former private key k_{priv} [15]. If a user A originally has $\left(k_{\mathrm{priv}_A}, k_{\mathrm{pub}_A}\right)$ as his keys, and another user B sends a message to A encrypted with k_{pub_A}, then if A refreshes his keys to a new pair $\left(k'_{\mathrm{priv}_A}, k'_{\mathrm{pub}_A}\right)$ while the message is on its way, then A might have decryption failures because the message was encrypted using his old public key. Additionally, each user refreshing his keys must perform a full key generation, which normally is computationally expensive [16]. Unlike traditional public key cryptosystems, the **O2MD2-I** framework provides a distributed key refresh capability. Every entity can perform a key refresh while preserving existing communication channels with other entities. This means that if a message is already on its way, it can be deciphered without failures by the recipient even though it was originally encrypted with the recipient's old public key. This is extremely helpful because every user can refresh their keys frequently, avoiding attacks aiming to guess their private key based on the public key. The distributed key refresh runs in polynomial time, and can be implemented on devices where computation time and space are major constraints, as it is with Internet of Things (IoT) devices.

We also provide twelve suggested configurations which vary on their memory requirements and their provided security level. Six implementations are designed for 64-bit architectures, while the other six for 32-bit architectures. The six implementations in each architecture are classified as Low, Middle, and High security, which are equivalent the

NIST Post-Quantum Cryptography Standardization project's security levels 1, 3 and 5, respectively. Finally, six patents of our framework describing all the previous characteristics have been filed and published in Taiwan, China, the United States, Japan, South Korea and the European Union [17].

In the next section we discuss some of the related work that has been explored. Section III and IV formally describe our solution and each framework with their key generation, encryption and decryption algorithms. Section V contains all the theorems and formulas related to the frameworks' proofs of correctness. Section VI presents a robust security analysis, performs tests for indistinguishability against random noise using the NIST *Statistical Test Suite*, formally proves the message integrity and authenticity verification capabilities of **O2MD2-II** and **O2MD2-III**, discusses about the security against brute force attacks, and gives additional comments. Section VII covers the speed test results, and Section VIII presents some conclusions and plans for future work.

II. RELATED WORK

The National Institute of Standards and Technology (NIST) started the *Post-Quantum Cryptography Standardization* project, aiming to identify, to analyze, and to publish a list of new quantum-resistant cryptosystems so they can be publicly tested. This project has multiple candidates listed under two categories: *Public-key Encryption and Key-establishment Algorithms*, and *Digital Signature Algorithms* [18]. Since its announcement, two classification rounds have been performed. Almost every algorithm in the second round has a public log of all the external reviewers' comments, concerns and suggestions, and the authors' official replies to each one of them [19]. Each algorithm also has a downloadable repository with their implementations in *c* language, which follows the *ECRYPT Benchmarking of Cryptographic Systems (eBACS)* interface proposed by the *Virtual Application and Implementation Research Lab (VAMPIRE)* [20], [21]. Besides these repositories, the Open Quantum Safe Organization created the open source *liboqs* project, whose goal is to provide development tools using the proposals submitted to the NIST project [22]. Even though the liboqs project is a recollection of the NIST proposals, it gets frequently outdated, as new versions of the algorithms are directly published on the NIST project's website [19].

Some of the candidates under the *Public-key Encryption and Key-establishment Algorithms* classification are NTRU [10], NTRU Prime [11], FrodoKEM [12], Crystals-Kyber [13], ThreeBears [14], LAC [23] and NewHope [24]. They differ in their security bases and used algebraic structures. For example, NTRU and NTRU Prime are lattice-based systems, where NTRU uses the ring $\mathbb{Z}_q[x]/\langle x^p - 1\rangle$ with q a power of 2, while NTRU Prime uses the ring $\mathbb{Z}_q[x]/\langle x^p - x - 1\rangle$. For both systems, p is prime [10], [11]. For Crystals-Kyber, it is based on the module learning with errors (MLWE) problem, and it uses the ring $\mathbb{Z}_{7681}[x]/\langle x^{256} + 1\rangle$ [25]. For ThreeBears, it is based on the integer module learning with errors (I-MLWE) problem, and it does not use cyclotomic polynomials. Instead, it uses the ring $\mathbb{Z}/N\mathbb{Z}$ with $N = q^{312} - q^{156} - 1$ and $q = 2^{10}$ [14]. Contrary to the previous four candidates, FrodoKEM does not use any ring at all, and relies on the learning with errors (LWE) problem [12]. LAC and NewHope are based on the ring learning with errors (RLWE) problem, but they need Number Theoretic Transforms (NTT) to enable fast computations [23], [24]. A detailed comparison of these cryptosystems is presented in Table 1.

Besides their security bases and algebraic structures, some of these algorithms require specialized arithmetic operations. For example, NTRU and NTRU Prime require polynomial center-lifts when applying modulo reduction [11], [31]. Crystals-Kyber, LAC and NewHope require NTT to efficiently compute polynomial multiplications over a ring. This impacts on the hardware, as native complex numbers support is needed [25], [36]. Other limitations are also known. For example, FrodoKEM uses the learning with errors (LWE) problem and not any ring-like structure, which forces it to use big-sized public keys [26]. Its authors also claim that it is IND-CCA secure, but a great number of reviewers have commented that the security proof presented in the NIST project is flawed [28]. For ThreeBears, the integer module learning with errors (I-MLWE) problem has not been studied as extensively as the learning with errors (LWE) problem nor the ring learning with errors (RLWE) problem [14]. For NewHope, there are public concerns on the reuse of the private keys in the CPA version, and for LAC, the key generation and encryption algorithms may leak information on timing attacks [23], [34], [35]. Finally, ThreeBears, NTRU, and NTRU Prime accept that it is difficult to verify all possible weaknesses, and it is unknown if vulnerabilities exist that are still undiscovered [14], [30]– [49]. Despite all the advantages and limitations of these candidates, they have been well received and acclaimed as innovative, and they are highly appreciated by the cryptographic community because they are pioneers in the search of post-quantum robust solutions.

A. ABOUT THE RING LEARNING WITH ERRORS (RLWE) PROBLEM

The ring learning with errors (RLWE) problem is a ring-based variation of the learning with errors (LWE) problem [37]. As with other variations of the learning with errors (LWE) problem, it is also considered to be computationally safe against quantum computers. Let $\{a_i(x)\}$ be a set of random known polynomials from the polynomial quotient ring $\mathbb{Z}_q[x]/\langle\phi(x)\rangle$ with $\phi(x)$ a polynomial, $\{e_i(x)\}$ be a set of random unknown polynomials in $\mathbb{Z}_q[x]/\langle\phi(x)\rangle$, and $s(x)$ be an unknown polynomial also in $\mathbb{Z}_q[x]/\langle\phi(x)\rangle$. Given the set $\{(a_i(x), e_i(x))\}$ and $s(x)$, then polynomials of the form

$$b_i(x) = a_i(x)s(x) + e_i(x) \tag{1}$$

are constructed. Two variations of the ring learning with errors (RLWE) problem are defined. The *Decision version* gives to a challenger a list of polynomials $\left\{\tilde{b}_i(x)\right\}$, and asks him to decide which ones were constructed following (1)

TABLE 1. **Comparison of the O2MD^2 cryptosystem against different post-quantum candidates in the NIST *Post-Quantum Cryptography Standardization* project (Round 2). Cryptosystems using different security bases were selected, and representative advantages and limitations are mentioned for each algorithm.**

Cryptography	Fundamental security problem	Algorithm	Characteristics		Post-quantum attacks
			Advantages	Limitations	
Classical	Integer Factorization	RSA, DSA	- Well studied. - Easy to implement in software. - Flexible.	- Requires modular exponentiation. - It is unknown if the Integer Factorization problem is NP-complete.	Weak against Shor's algorithm attacks.
	Discrete Logarithm Problem	ECC	- Well studied.	- Computationally hard additions.	Weak against Shor's and Grover's algorithm attacks.
	Unknown	AES	- Already embedded in Intel CPUs. - Easy to use in software modules.	- Unknown security basis problem. - Not broken yet.	
Post-Quantum	Learning with Errors (LWE)	FrodoKEM	- Easy to implement [26]. - Easy to escalate in software [27].	- Larger public keys compared to RSA, ECC and RLWE-based systems [26]. - Unclear IND-CCA security [9], [28]. - Requires uniform Gaussian distribution generators [29], [27]. - Implementation requires heavy memory utilization [27].	Strong against Shor's and Grover's algorithms attacks.
	Module Learning with Errors (MLWE)	Crystals-Kyber	- It is IND-CCA2 secure [25]. - It is easy to implement [25]. - It is easy to escalate [25].	- Requires NTT-capable hardware if implemented in hardware [25], [13]. - Needed Heavy software modifications if NTT is not supported [25]. - Already known *Fault Attacks* [25].	
	Integer Module Learning with Errors (I-MLWE)	ThreeBears	- Fast key generation [14]. - Public keys have small sizes [14]. - Does not need NTT.	- Requires Mersenne primes [14]. - I-MLWE has not been studied as extensively as LWE or RLWE [14]. - Difficult analysis of failure [14] [30]. - Possible weakness against side-channel attacks [14].	
	Lattice-based Encryption	NTRU	- Well-studied algebra [31]. - Flexible and adaptable [31]. - Simple to implement, as it only needs two parameters [31].	- Security has not been tested enough [31]. - Big-sized keys are needed [31]. - Lack of strict structure [31] [10].	
		NTRU Prime	- Strong against homomorphism-based attacks [32]. - Fast performance [32]. - Secure even if Galois groups are broken [32].	- Vulnerabilities might still be undiscovered [32]. - Difficult security analysis [32], [33].	
	Ring Learning with Errors (RLWE)	NewHope	- High performance [24]. - Memory efficient. - Strong even after possible leak of information [24].	- It either provides NIST security levels 1 or 5, not 2, 3 and 4 [24]. - Requires NTT [24]. - Public concerns on reuse of the private keys in CPA version [34].	
		LAC	- Fast performance on Intel x64 and ARM (NEON) CPUs [23]. - Parallel design, suitable for multi-core CPUs [23]. - Simple design (small modulo) [23].	- Cannot be sped up (even by NTT) [23]. - Expensive overhead for error correction in decryption [23]. - KeyGen and Encrypt may leak information on timing attacks [23], [35]. - Fix for timing attacks affects computation efficiency [23].	
		O2MD2	- Hardware-efficient. - Fast performance. - Three designs with different levels of security. - No decryption errors. - Message Integrity and Authenticity verifications.	- Still unknown all possible attacks and vulnerabilities. - Requires software or hardware capable of performing polynomial rings algebra operations. - Requires hash functions.	

and which ones were randomly generated. The *Search version* asks to calculate the polynomial $s(x)$ given a list of polynomial pairs $\{(a_i(x), b_i(x))\}$. In an average case scenario, the ring learning with errors (RLWE) problem is known to be at least NP-hard, and it is reducible into the approximate shortest vector problem (α-SVP) [37].

The ring learning with errors (RLWE) problem has two essential key points: First, as the polynomials are defined in a ring, the addition and multiplication can be performed in polynomial time. Second, the core of the ring learning with errors (RLWE) problem is the indistinguishability of the generated polynomials against randomly generated ones, which is advantageous when designing a secure cryptosystem.

III. PROBLEM STATEMENT AND SOLUTION OVERVIEW

Our main goal is to create a quantum-resistant public key cryptosystem based on the ring learning with errors (RLWE) problem and provides a one-private key / multiple-public keys architecture (i.e., from one single private key, multiple public keys can be generated). It should also perform a distributed public key refresh which does not affect existing communication channels with other users, and its encryption and decryption protocols should run in polynomial time and space. Finally, it should have a strong security level, and provide message integrity and message authentication verifications.

A. OVERVIEW OF OUR SOLUTION

Our proposal explores the use of arithmetic functions as seeds of information, and use them to construct custom structures that we call p-bases which are isomorphic to $\mathbb{Z}^m$. By selecting a random p-base, we define as b to its maximum component. Later with a random prime p_1 and b, we calculate a second prime p_2. With b, p_1 and p_2, we construct three rings isomorphic to $\mathbb{Z}_{b+1}[x]/\langle x^m - 1\rangle$, $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$, where the polynomial ring addition and product are performed.

Our solution provides a one-to-many private/public key architecture. We create the concept of **sessions**, where each session will allow one private key to generate multiple public ones. This provides extra security against attacks aiming to guess the private key. Our framework also has a distributed key refresh mechanism. This refresh does not interfere with previous communications, so when an entity A performs a key refresh obtaining new keys, messages encrypted with the old public key that are still on their way can be decrypted without problems by the new key. This is extremely convenient for networks where messages must pass through multiple hops and networks with high latency, like the case of Internet of Things (IoT) networks, where a centralized entity in charge of coordinating the key refresh process might be difficult to implement. Any entity B who wishes to send a message to A does not need to retrieve A's public key every time. B can store A's public key once, and keep using it without worries of decryption failures.

Over and above the previous features, our cryptosystem also shows how to generate persistent information from an arithmetic function. This feature is ideal for low-level hardware, where memory is a major constraint. Given an arithmetic function (which could be embedded in hardware), we create custom data structures called **instances**. For a particular instance, the pair of the arithmetic function and the calculated instance will work as our private key. One single arithmetic function can generate innumerable instances, and they can be calculated in polynomial time. This provides the ability to generate multiple private keys from a single function. By saving the data of that instance, the same private key can be obtained in the future. The ability of generating multiple private/public keys offers advantages against traditional private keys: as stored-in-memory private keys are normally static, they occupy memory which might be a constrained resource in the device. Besides this, if an attacker guesses a stored-in-memory key, a firmware update would be needed to change the compromised private key.

Finally, our solution offers three different frameworks that provide increasing security capabilities. Messages integrity verifications are performed in order to verify if a ciphertext was modified prior to its reception (by either noise or on purpose), and messages authentication verifications allow to determine if the sender of a message is its creator or not.

In the following sections we introduce the concept of p-bases and we make the formal description for our cryptosystem. Later we analyze its security and evaluate the produced ciphertexts using the NIST *Statistical Test Suite*. Finally, we compare the performance of our solution against nineteen different implementations from the NIST *Post-Quantum Cryptography Standardization* project.

B. PRELIMINARIES

Let $f : \mathbb{N} \to \mathbb{N}$ be an arithmetic function whose outputs are uniformly distributed throughout $\mathbb{N}$. We define the associated matrix of f as

$$[f] = \begin{pmatrix} f(2^0) & f(2^1) & f(2^2) & \dots \\ f(3^0) & f(3^1) & f(3^2) & \dots \\ f(5^0) & f(5^1) & f(5^2) & \dots \\ \vdots & \vdots & \vdots & \dots \end{pmatrix} \tag{2}$$

where the n-th row contains all the images of the powers of the n-th prime, i.e., each row is of the form

$$\overrightarrow{f}_p = \left[f(p^0), f(p^1), f(p^2), f(p^3), \dots \right], \tag{3}$$

where p is a prime. For prime p and positive integers s, t, we define an **instance** as $I = (p, s, t)$, and for an instance I we define the p-base of size m of f as

$$\overleftrightarrow{f}_p\Big|_{s,t}^{m} = \sum_{i=0}^{t} \left[f\left(p^{s+im}\right), \dots, f\left(p^{s+im+(m-1)}\right) \right], \tag{4}$$

i.e., for $\overrightarrow{f}_p$, starting from position s, we add t vectors of m consecutive values. For example, for f defined as

$$f(p^n) = \text{the } n\text{-th digit in the fractional part of } \sqrt{p}, \text{ for } n > 0, \tag{5}$$

and $f(1) = 1$, we have that

$$[f] = \begin{pmatrix} 1 & 4 & 1 & 4 & 2 & 1 & 3 & \dots \\ 1 & 7 & 3 & 2 & 0 & 5 & 0 & \dots \\ 1 & 2 & 3 & 6 & 0 & 6 & 7 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{6}$$

then by picking $I = (p, s, t) = (3, 0, 1)$ and $m = 5$ we have

$$\begin{aligned} \overleftrightarrow{f}_3\Big|_{0,1}^{5} &= [1+5, 7+0, 3+8, 2+0, 0+7] \\ &= [6, 7, 11, 2, 7]. \end{aligned} \tag{7}$$

The use of this notation has three main advantages: First, any device that has f embedded in it can retrieve the same base by saving I and m. This is useful for devices where memory is a constraint, as it is for Internet of Things (IoT) devices. Second, by changing the values of I, we can generate multiple bases. This is the first step towards building a one-to-many private/public key architecture. Third, if an attacker knows both the function f and a list of bases $\left\{ \overleftrightarrow{f}_p\Big|^m_{s,t} \right\}$, it would be computationally difficult to guess the instances $\{I\}$ that generated them, as the attacker would need a polynomial time oracle to solve the subset sum problem, which is NP-complete (and, even more, most approaches to solve it by quantum computers require exponential time) [38].

We define the concept of public **session** $\mathbb{S}$ as an ordered set of integers and one-way hash functions that will provide with parameters to configure the whole framework. We say that a user joins the session if he uses $\mathbb{S}$ when generating his keys and when communicating with other users. For any party of participants, as long as they joined the same session, they will be able to communicate. On the other hand, a user may join into multiple sessions at the same time. Hence one user with one arithmetic function can simultaneously join multiple sessions and generate multiple private/public keys for each one of those sessions.

For any two bases of size m, we use the usual addition and multiplication from the quotient ring $\mathbb{Z}[x]/\langle x^m - 1\rangle$, denoted as $+$ and $\circledast$, respectively. As the instance I should be kept secret, we denote a p-base of size m as $\overleftrightarrow{f}\Big|^m$. For primes p_1 and p_2, we denote the multiplicative inverse of $\overleftrightarrow{f}\Big|^m$ in $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$ as $\overleftrightarrow{F}_{p_1}\Big|^m$ and $\overleftrightarrow{F}_{p_2}\Big|^m$, respectively. In case $\overleftrightarrow{f}\Big|^m$ is not invertible in $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ or $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$, we select a different p_1 or p_2.

As an example, let $\overleftrightarrow{f}\Big|^5 = [2, 81, 27, 9, 3]$ be result of a secret function f and a secret instance I, and let $p_1 = 251$ and $p_2 = 18072001$ be two primes. Then

$$\begin{aligned} \overleftrightarrow{F}_{251}\Big|^5 &= [92, 223, 74, 164, 128], \\ \overleftrightarrow{F}_{18072001}\Big|^5 &= [11798464, 16030112, 7407741, \\ &\quad 1287507, 11026277], \end{aligned} \tag{8}$$

where

$$\begin{aligned} \overleftrightarrow{f}\Big|^5 \circledast \overleftrightarrow{F}_{251}\Big|^5 &= [17821, 16315, 13052, 22339, 13555] \\ &= [0, 0, 0, 0, 1] (\text{mod } 251), \end{aligned} \tag{9}$$

and

$$\begin{aligned} &\overleftrightarrow{f}\Big|^5 \circledast \overleftrightarrow{F}_{18072001}\Big|^5 \\ &\quad = [506016028, 1066248059, \\ &\qquad 1319256073, 1734912096, 1174680066] \\ &\quad = [0, 0, 0, 0, 1] (\text{mod } 18072001). \end{aligned} \tag{10}$$

For any base $\overleftrightarrow{f}\Big|^m$, the notation $\overleftrightarrow{f}\Big|^m_{[i,j]}$ will denote the vector

$$\overleftrightarrow{f}\Big|^m_{[i,j]} = \left[\overleftrightarrow{f}\Big|^m (i),\ \overleftrightarrow{f}\Big|^m (i+1), \ldots,\ \overleftrightarrow{f}\Big|^m (j) \right], \tag{11}$$

with $1 \le i \le j \le m$. Finally, for two bases $\overleftrightarrow{f}\Big|^{m_1}$ and $\overleftrightarrow{g}\Big|^{m_2}$ (of sizes m_1 and m_2, respectively), the concatenation

TABLE 2. Notation table.

f	Uniformed distributed outputs arithmetic function.
$\mathbb{S}$	Session.
$I = (p, s, t)$	Instance.
$\overleftrightarrow{f}_p\Big\|^m_{s,t}$	p-base of f of order m.
$\overleftrightarrow{f}\Big\|^m + \overleftrightarrow{g}\Big\|^m$	Addition.
$\overleftrightarrow{f}\Big\|^m \circledast \overleftrightarrow{g}\Big\|^m$	Convoluted multiplication.
$\overleftrightarrow{F}_\ell\Big\|^m$	Inverse of $\overleftrightarrow{f}\Big\|^m$ modulo ℓ.
$\overleftrightarrow{f}\Big\|^m_{[i,j]}$	Segment of $\overleftrightarrow{f}\Big\|^m$ from i to j.
$\overleftrightarrow{f}\Big\|^{m_1} \oplus \overleftrightarrow{g}\Big\|^{m_2}$	Concatenation of $\overleftrightarrow{f}\Big\|^{m_1}$ and $\overleftrightarrow{g}\Big\|^{m_2}$.

$\overleftrightarrow{f}\Big|^{m_1} \oplus \overleftrightarrow{g}\Big|^{m_2}$ will be denoted as

$$\begin{aligned} \overleftrightarrow{f}\Big|^{m_1} \oplus \overleftrightarrow{g}\Big|^{m_2} = \Big[&\overleftrightarrow{f}\Big|^{m_1} (1), \ldots, \overleftrightarrow{f}\Big|^{m_1} (m_1), \\ &\overleftrightarrow{g}\Big|^{m_2} (1), \ldots, \overleftrightarrow{g}\Big|^{m_2} (m_2) \Big], \end{aligned} \tag{12}$$

where $\overleftrightarrow{f}\Big|^{m_1} \oplus \overleftrightarrow{g}\Big|^{m_2}$ is an array of size $m_1 + m_2$. All these notation conventions are presented in Table 2.

IV. SOLUTION

Our proposal is the **O2MD^2 system**, consisting of three frameworks: **O2MD2-I** (KeyGen-I, KeySoftReset-I, Encrypt-I and Decrypt-I), **O2MD2-II** (KeyGen-II, Encrypt-II and Decrypt-II), and **O2MD2-III** (KeyGen-III, Encrypt-III and Decrypt-III). The security and capabilities provided by each framework increase when compared to its predecessor. The three frameworks share a sub-algorithm called *Randomization*, described in Algorithm 1. This algorithm requires to use a discrete Gaussian sampling over $\mathbb{Z}$ in order to protect the messages and the generated keys. Similarly, each framework requires slightly different public session variables $\mathbb{S}$, which provide to each user with the parameters needed to perform a correct communication. In the following subsections, we describe each framework, and in the next section we perform a proof of correctness for each one of them.

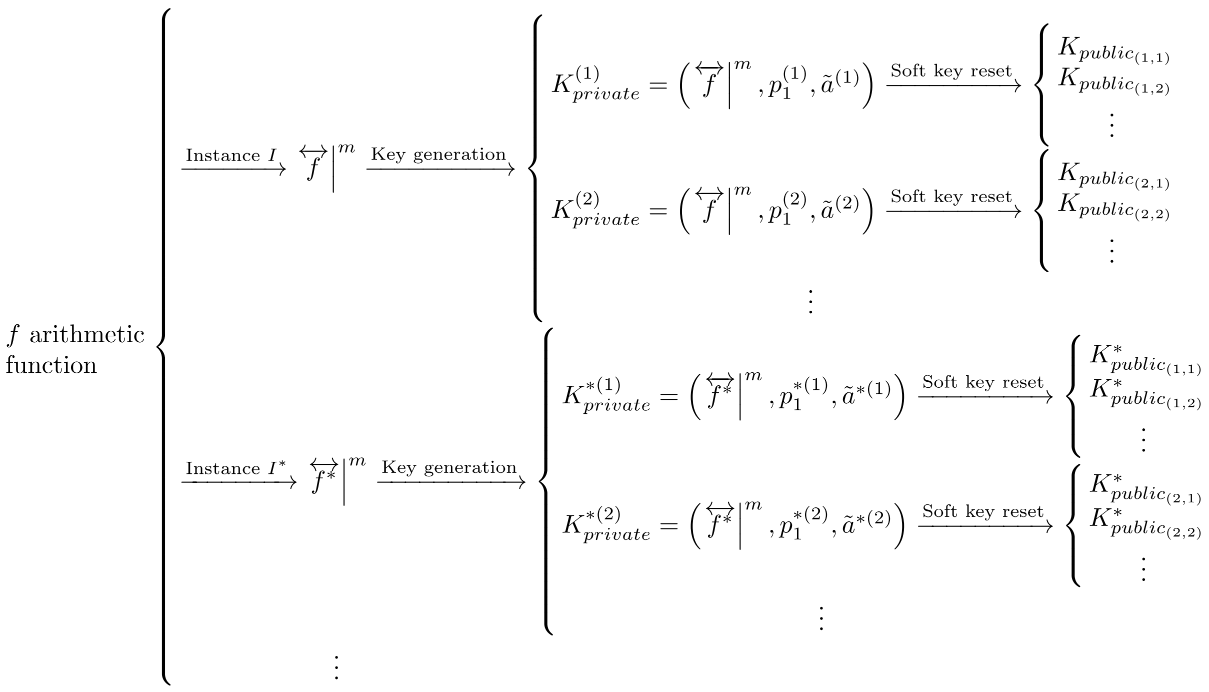
There are three reasons why we define the randomization operation as its own: First, this operation helps us to add randomness in our protocols. Second, it is computationally difficult for an attacker to recover the inputs of this operation based on the components of the output. Finally, we use this function as the soft key-reset algorithm for our public keys.

A. O2MD2-I

This framework requires a public session of the form

$$\mathbb{S} = (m, \tilde{b}, r), \tag{32}$$

where m, $\tilde{b}$ and r are positive integers, for which m describes the size of the arrays to use for all users, $\tilde{b}$ is a constant used for performing perfect decryption for all users, and r is a constant used to let all the users know the maximum size of the alphabet to use for the messages. Four algorithms are proposed: key generation, soft key-reset, encryption and decryption.



f arithmetic function
Instance I
$\overleftrightarrow{f}\big|^m$
Key generation
$K^{(1)}_{private} = \left(\overleftrightarrow{f}\big|^m, p_1^{(1)}, \tilde{a}^{(1)}\right)$
Soft key reset
$K_{public_{(1,1)}}$
$K_{public_{(1,2)}}$
$K^{(2)}_{private} = \left(\overleftrightarrow{f}\big|^m, p_1^{(2)}, \tilde{a}^{(2)}\right)$
$K_{public_{(2,1)}}$
$K_{public_{(2,2)}}$
Instance I^*
$\overleftrightarrow{f^*}\big|^m$
$K^{*(1)}_{private} = \left(\overleftrightarrow{f^*}\big|^m, p_1^{*(1)}, \tilde{a}^{*(1)}\right)$
$K^*_{public_{(1,1)}}$
$K^*_{public_{(1,2)}}$
$K^{*(2)}_{private} = \left(\overleftrightarrow{f^*}\big|^m, p_1^{*(2)}, \tilde{a}^{*(2)}\right)$
$K^*_{public_{(2,1)}}$
$K^*_{public_{(2,2)}}$

FIGURE 1. O2MD2-I Key generation and soft key-reset working together. A single arithmetic function and multiple instances ($I, I^*, \ldots$) can generate multiple private keys via the key generation algorithm. For a fixed private key, the soft key-reset generates multiple public keys in polynomial time. As long as the private key remains fixed, all messages encrypted with any public key generated by that specific private key can be decrypted.

TABLE 3. Comparison of the three O2MD^2 frameworks. The first column shows the requirements of OW (one-way) hash functions in the session. The second column shows the amount of hash functions needed to be used. The third column shows if the framework performs message integrity checks or not. The fourth column shows if the key generation creates signature keys, and the fifth column shows if the framework performs any message authenticity check.

Framework	Requires OW hash functions	Number of hash functions	Performs Message Integrity Check	Has Signature Key	Performs Message Authenticity Check
O2MD2-I	No	0	No	No	No
O2MD2-II	Yes	1	Yes	No	No
O2MD2-III	Yes	2	Yes	Yes	Yes

Algorithm 1 (O2MD2) Randomization. *(This Function is Called by* ***(O2MD2-I)*** *Soft Key-Reset and Encryption, and Both* ***(O2MD2-II)*** *and* ***(O2MD2-III)*** *Key Generation).*

```
input : Array $\overleftrightarrow{h}\big|^m$, positive integers $x$ and $y$.
output: Array $\overleftrightarrow{h}_{\text{random}}\big|^m$.
Let $\overleftrightarrow{R}\big|^m_{(y)}$ be a $m$-sized vector where each element is sampled from a discrete Gaussian distribution on $\mathbb{Z}$ and reduced modulo $y$.
Return
```

$$\overleftrightarrow{h}_{\text{random}}\Big|^m = x\left(\overleftrightarrow{h}\Big|^m \circledast \overleftrightarrow{R}\Big|^m_{(y)}\right) \tag{13}$$

1) KEY GENERATION

The key generation is shown in Algorithm 2. It takes the session $\mathbb{S} = (m, \tilde{b}, r)$, an instance $\overleftrightarrow{f}\big|^m$ of f, a random positive integer $\tilde{a}$ and a random prime p_1 as input, with

$$\tilde{r} = \max(\tilde{b}, r) < p_1. \tag{33}$$

Define b as

$$b = max(\overleftrightarrow{f}\big|^m), \tag{34}$$

and let p_2 be any random prime holding

$$p_2 > \max(p_1 m\tilde{a}\tilde{b}, mb\tilde{r}). \tag{35}$$

Algorithm 2 (O2MD2-I) Key Generation.

```
input : Instance $\overleftrightarrow{f}\big|^m$, session $\mathbb{S} = (m, \tilde{b}, r)$, random
        prime $p_1$ with $\tilde{r} = \max(\tilde{b}, r) < p_1$, and random
        positive integer $\tilde{a}$.
output: Message: *Cannot construct keys based on the
        inputs*, or the set of keys
        $K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\big|^m, p_2\right)$ and
        $K_{\text{private}} = \left(\overleftrightarrow{f}\big|^m, p_1, \overleftrightarrow{F}_{p_1}\big|^m, \overleftrightarrow{F}_{p_2}\big|^m, \tilde{a}\right)$.
1 if $\overleftrightarrow{f}\big|^m$ contains at least one negative component then
2     Return Cannot construct keys based on the inputs.
3 end
4 Calculate
      $b = \max\left(\overleftrightarrow{f}\big|^m\right)$,
      $\tilde{r} = \max\left(\tilde{b}, r\right)$.        (14)
5 Select a random prime $p_2$ such that
      $p_2 > \max(p_1 m\tilde{a}\tilde{b}, mb\tilde{r})$.        (15)
6 if $\overleftrightarrow{F}_{p_1}\big|^m$ or $\overleftrightarrow{F}_{p_2}\big|^m$ does not exist then
7     Select a different prime $p_1$ or $p_2$, respectively.
8     Restart Key Generation.
9 end
10 Calculate
      $\overleftrightarrow{K}_{\text{public}}\big|^m = \text{Soft key reset}\left(\overleftrightarrow{F}_{p_2}\big|^m, p_1, p_2, \tilde{a}\right)$.        (16)
11 Return
      $K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\big|^m, p_2\right)$,
      $K_{\text{private}} = \left(\overleftrightarrow{f}\big|^m, p_1, \overleftrightarrow{F}_{p_1}\big|^m, \overleftrightarrow{F}_{p_2}\big|^m, \tilde{a}\right)$.        (17)
```

Algorithm 3 (O2MD2-I) Soft key-reset.

```
input : Inverse $\overleftrightarrow{F}_{p_2}\big|^m$, primes $p_1, p_2$ and positive integer $\tilde{a}$.
output: Public key $\overleftrightarrow{K}_{\text{public}}\big|^m$.
1 Return
      $\overleftrightarrow{K}_{\text{public}}\big|^m = \text{Randomization}\left(\overleftrightarrow{F}_{p_2}\big|^m, p_1, \tilde{a}\right) \pmod{p_2}$.        (18)
```

Algorithm 4 (O2MD2-I) Encryption.

```
input : $K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\big|^m, p_2\right)$, session $\mathbb{S} = (m, \tilde{b}, r)$ and Message $\overleftrightarrow{M}\big|^m$.
output: Ciphertext $\overleftrightarrow{Cipher}\big|^m$.
1 Calculate
      $\overleftrightarrow{R}\big|^m = \text{Randomization}\left(\overleftrightarrow{K}_{\text{public}}\big|^m, 1, \tilde{b}\right)$.        (19)
2 Return
      $\overleftrightarrow{Cipher}\big|^m = \left(\overleftrightarrow{M}\big|^m + \overleftrightarrow{R}\big|^m\right) \pmod{p_2}$.        (20)
```

The inequalities (33) and (35) guarantee a correct decryption of the ciphertexts. If the selected $\overleftrightarrow{f}\big|^m$ is not invertible in $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ or $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$, we select different random primes p_1 and p_2 satisfying both inequalities. After finally selecting a pair of valid primes, we run the soft key-reset algorithm (described below) over $\overleftrightarrow{F}_{p_2}\big|^m, p_1, p_2$ and $\tilde{a}$, which returns the public key $\overleftrightarrow{K}_{\text{public}}\big|^m$.

Finally, we group $\overleftrightarrow{K}_{\text{public}}\big|^m$ and p_2 as the public key, and $\overleftrightarrow{f}\big|^m, p_1, \overleftrightarrow{F}_{p_1}\big|^m, \overleftrightarrow{F}_{p_2}\big|^m$ and $\tilde{a}$ as the private key, returning the following set of keys.

$$K_{\text{public}} = (\overleftrightarrow{K}_{\text{public}}\big|^m, p_2),$$
$$K_{\text{private}} = (\overleftrightarrow{f}\big|^m, p_1, \overleftrightarrow{F}_{p_1}\big|^m, \overleftrightarrow{F}_{p_2}\big|^m, \tilde{a}). \quad (36)$$

For two parties A and B, we will denote their public and private keys as

$$K_{\text{public}_A} = (\overleftrightarrow{K}_{\text{public}_A}\big|^m, p_2),$$
$$K_{\text{private}_A} = (\overleftrightarrow{f}\big|^m, p_1, \overleftrightarrow{F}_{p_1}\big|^m, \overleftrightarrow{F}_{p_2}\big|^m, \tilde{a}) \quad (37)$$

Algorithm 5 (O2MD2-I) Decryption.

```
input : Ciphertext $\overleftrightarrow{Cipher}\Big|^{m}$, prime $p_2$, inverse $\overleftrightarrow{F}_{p_1}\Big|^{m}$ and $K_{\text{private}} = \left(\overleftrightarrow{f}\Big|^{m}, p_1, \overleftrightarrow{F}_{p_1}\Big|^{m}, \overleftrightarrow{F}_{p_2}\Big|^{m}, \tilde{a}\right)$.
output: Message $\overleftrightarrow{M_1}\Big|^{m}$.
1 Calculate $\overleftrightarrow{M_0}\Big|^{m}$ as
      $\overleftrightarrow{M_0}\Big|^{m} = \left[\left(\overleftrightarrow{Cipher}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m}\right) (\bmod\, p_2)\right] (\bmod\, p_1).$   (21)
2 Return
      $\overleftrightarrow{M_1}\Big|^{m} = \overleftrightarrow{M_0}\Big|^{m} \circledast \overleftrightarrow{F}_{p_1}\Big|^{m} (\bmod\, p_1).$   (22)
```

Algorithm 6 (O2MD2-II) Key generation.

```
input : Instance $\overleftrightarrow{f}\Big|^{m}$, session $\mathbb{S} = (m, \tilde{b}, r, H)$ with $2 \mid m$,
        random prime $p_1$ with $\tilde{r} = \max(\tilde{b}, r) < p_1$, and
        random positive integer $\tilde{a}$.
output: Message: Cannot construct keys based on the
        inputs, or the set of keys
        $K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\Big|^{m}, p_2\right)$ and
        $K_{\text{private}} = \left(\overleftrightarrow{f}\Big|^{m}, p_1, \overleftrightarrow{F}_{p_1}\Big|^{m}, \overleftrightarrow{F}_{p_2}\Big|^{m}, \tilde{a}\right)$.
1 if $\overleftrightarrow{f}\Big|^{m}$ contains at least one negative component then
2   | Return Cannot construct keys based on the inputs.
3 end
4 Calculate
      $b = \max\left(\overleftrightarrow{f}\Big|^{m}\right),$
      $\tilde{r} = \max\left(\tilde{b}, r\right).$   (23)
5 Select a random prime $p_2$ such that
      $p_2 > \max(p_1 m \tilde{a} \tilde{b}, m b \tilde{r}).$   (24)
6 if $\overleftrightarrow{F}_{p_1}\Big|^{m}$ or $\overleftrightarrow{F}_{p_2}\Big|^{m}$ do not exist then
7   | Select a different prime $p_1$ or $p_2$, respectively. Restart
    | Key Generation.
8 end
9 Calculate
      $\overleftrightarrow{K}_{\text{public}}\Big|^{m} = \textbf{Randomization}\left(\overleftrightarrow{F}_{p_2}\Big|^{m}, p_1, \tilde{a}\right)(\bmod\, p_2).$   (25)
10 Return
      $K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\Big|^{m}, p_2\right),$
      $K_{\text{private}} = \left(\overleftrightarrow{f}\Big|^{m}, p_1, \overleftrightarrow{F}_{p_1}\Big|^{m}, \overleftrightarrow{F}_{p_2}\Big|^{m}, \tilde{a}\right).$   (26)
```

Algorithm 7 (O2MD2-II) Encryption.

```
input : $K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\Big|^{m}, p_2\right)$, session
        $\mathbb{S} = (m, \tilde{b}, r, H)$ and Message $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$.
output: Ciphertext $\overleftrightarrow{Cipher}\Big|^{m}$.
1 Let $\overleftrightarrow{R}\Big|^{\frac{m}{2}}$ be a $\frac{m}{2}$-sized vector where each element is
  sampled from a discrete Gaussian distribution on $\mathbb{Z}$ and
  reduced modulo $r$.
2 Calculate
      $\overleftrightarrow{M^*}\Big|^{m} = \overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R}\Big|^{\frac{m}{2}},$
      $\overleftrightarrow{H_{M^*}}\Big|^{m} = H\left(\overleftrightarrow{M^*}\Big|^{m}\right).$   (27)
3 Return
      $\overleftrightarrow{Cipher}\Big|^{m} = \left(\overleftrightarrow{K}_{\text{public}}\Big|^{m} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m} + \overleftrightarrow{M^*}\Big|^{m}\right)(\bmod\, p_2).$   (28)
```

for A, and

$$\begin{aligned} K_{\text{public}_B} &= \left(\overleftrightarrow{K}_{\text{public}_B}\Big|^{m}, q_2\right), \\ K_{\text{private}_B} &= \left(\overleftrightarrow{g}\Big|^{m}, q_1, \overleftrightarrow{G}_{q_1}\Big|^{m}, \overleftrightarrow{G}_{q_2}\Big|^{m}, \tilde{\alpha}\right) \end{aligned} \tag{38}$$

for B, where p_1 and p_2 are the selected primes for A, and q_1 and q_2 the selected primes for B.

Finally, it is worth to mention that the private key shown in (36) can be shorter. The alternative private keys

$$K_{\text{private}} = \left(\overleftrightarrow{f}\Big|^{m}, p_1, \tilde{a}\right), \tag{39}$$

Algorithm 8 (O2MD2-II) Decryption.

```
input : Ciphertext $\overleftrightarrow{Cipher}\Big|^{m}$, session $\mathbb{S} = (m, \tilde{b}, r, H)$, $K_{\text{public}} = (\overleftrightarrow{K}_{\text{public}}\Big|^{m}, p_2)$, inverse $\overleftrightarrow{F}_{p_1}\Big|^{m}$ and
        $K_{\text{private}} = (\overleftrightarrow{f}\Big|^{m}, p_1, \overleftrightarrow{F}_{p_1}\Big|^{m}, \overleftrightarrow{F}_{p_2}\Big|^{m}, \tilde{a})$.
output: Message $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$.
1 Calculate $\overleftrightarrow{M_0}\Big|^{m}$, $\overleftrightarrow{M_1}\Big|^{m}$ and $\overleftrightarrow{H_{M_1}}\Big|^{m}$ as
      $\overleftrightarrow{M_0}\Big|^{m} = [(\overleftrightarrow{Cipher}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m})(\text{mod}\, p_2)](\text{mod}\, p_1),$
      $\overleftrightarrow{M_1}\Big|^{m} = \overleftrightarrow{M_0}\Big|^{m} \circledast \overleftrightarrow{F}_{p_1}\Big|^{m} (\text{mod}\, p_1),$        (29)
      $\overleftrightarrow{H_{M_1}}\Big|^{m} = H(\overleftrightarrow{M_1}\Big|^{m}).$
2 Calculate $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$ as
      $\overleftrightarrow{M'}\Big|^{\frac{m}{2}} = \overleftrightarrow{M_1}\Big|^{m}_{[1,\frac{m}{2}]}.$        (30)
3 Calculate $\overleftrightarrow{Cipher'}\Big|^{m}$ as
      $\overleftrightarrow{Cipher'}\Big|^{m} = (\overleftrightarrow{K}_{\text{public}}\Big|^{m} \circledast \overleftrightarrow{H_{M_1}}\Big|^{m} + \overleftrightarrow{M_1}\Big|^{m})(\text{mod}\, p_2).$        (31)
4 if $\overleftrightarrow{Cipher'}\Big|^{m} == \overleftrightarrow{Cipher}\Big|^{m}$ And $0 \leq \overleftrightarrow{M_1}\Big|^{m}(i) < r$, for $i = 1, \ldots, m$ then
5     Return $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$,
6 else
7     Return Invalid ciphertext received.
8 end
```

and

$$K_{\text{private}} = (f, I, p_1, \tilde{a}), \tag{40}$$

can be used for different scenarios. By only storing (f, I) or $\overleftrightarrow{f}\Big|^{m}$, the values of $\overleftrightarrow{F}_{p_1}\Big|^{m}$ and $\overleftrightarrow{F}_{p_2}\Big|^{m}$ can be calculated on-demand, but this would require a space-time trade-off. Either we store more information in the private key and avoid re-calculations, or we save memory by only storing either (f, I) or $\overleftrightarrow{f}\Big|^{m}$ but we would require additional time to re-calculate $\overleftrightarrow{F}_{p_1}\Big|^{m}$ and $\overleftrightarrow{F}_{p_2}\Big|^{m}$ when needed. The keys (39) and (40) can be useful in devices where memory storage is an issue, and where latency in the key generation is not a problem.

2) SOFT KEY-RESET

The soft key-reset algorithm produces an array that will work as the public key. It allows the users to generate a new public key in polynomial time without modifying the private key. This algorithm works in a distributed way, so no central entity is needed to coordinate a key refresh for all the entities in the network. This means that it can be executed on-demand, so any entity can perform the soft key-reset as needed. Finally, this algorithm provides backwards compatibility for decryption, meaning that if a message was encrypted using one public key and a ciphertext is received after performing multiple soft key-resets, as long as the public keys have been generated by the soft key-reset, the ciphertext can be deciphered correctly.

A diagram on how this algorithm is embedded in the key generation is shown in Fig. 1, and its formal description is shown in Algorithm 3.

3) ENCRYPTION

For our encryption algorithm, as all the users joined the same public session $\mathbb{S} = (m, \tilde{b}, r)$, we require all the plaintexts to be of the form $\overleftrightarrow{M}\Big|^{m} \in (\mathbb{Z}_r)^m$ with $0 \leq \overleftrightarrow{M}\Big|^{m}(i) < r$, for $i = 1, \ldots, m$. This algorithm is shown in Algorithm 4. As with the Soft key-reset, the encryption algorithm also uses the *Randomization* function which requires to sample from the discrete Gaussian distribution in order to protect the message.

4) DECRYPTION

Finally, the decryption algorithm shown in Algorithm 5 uses the private key and the received ciphertext to decrypt the message. As long as p_1 and p_2 satisfy inequalities (33) and (35), perfect decryption is always achieved.

B. O2MD2-II

This framework, unlike **O2MD2-I**, consist of three algorithms: key generation, encryption and decryption algorithm. It requires a public session of the form

$$\mathbb{S} = (m, \tilde{b}, r, H), \tag{41}$$

where m, $\tilde{b}$ and r follow the same rules and purposes as in the session for **O2MD2-I**, with the addition that $2 \mid m$. Additionally, the **O2MD2-II** public session requires a one-way hash function H of the form

$$H : \mathbb{Z}^m \to \left(\mathbb{Z}_{\tilde{b}}\right)^m. \tag{42}$$

The **O2MD2-II** framework provides a stronger security compared to **O2MD2-I**. By the use of H in its encryption and decryption stages, **O2MD2-II** provides ciphertext integrity check, so when modified ciphertexts are received, the decryption algorithm rejects them.

1) KEY GENERATION

The key generation for **O2MD2-II** is shown in Algorithm 6. It is similar to the key generation for **O2MD2-I**, but with the difference that the randomization function is called directly, so no soft key-reset is used in this framework. Additionally, both p_1 and p_2 follow the inequalities (33) and (35) as in **O2MD2-I**. Finally, for two parties A and B, we will denote their public and private keys using the same notation as shown in (37) and (38).

2) ENCRYPTION

The encryption algorithm for **O2MD2-II** (shown in Algorithm 7) requires all the users to use the same session $\mathbb{S}$ described in (41). All the plaintexts must be of the form $\overleftrightarrow{M}\Big|^{\frac{m}{2}} \in (\mathbb{Z}_r)^{\frac{m}{2}}$ with $0 \leq \overleftrightarrow{M}\Big|^{m}(i) < r$, for $i = 1, \ldots, \frac{m}{2}$. Before encrypting $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$, an $\frac{m}{2}$-sized vector $\overleftrightarrow{R}\Big|^{\frac{m}{2}}$ is selected, for which each element is sampled from a discrete Gaussian distribution on $\mathbb{Z}$ and reduced modulo r. Then the two vectors $\overleftrightarrow{M^*}\Big|^{m}$ and $\overleftrightarrow{H_{M^*}}\Big|^{m}$ are calculated as shown in Equation (27), where $\overleftrightarrow{M^*}\Big|^{m}$ is the concatenation of $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{R}\Big|^{\frac{m}{2}}$. Finally, $\overleftrightarrow{K}_{\text{public}}\Big|^{m}$ and $\overleftrightarrow{H_{M^*}}\Big|^{m}$ are used to encrypt $\overleftrightarrow{M^*}\Big|^{m}$.

3) DECRYPTION

The decryption algorithm for **O2MD2-II** is shown in Algorithm 8. It uses the private key, the session $\mathbb{S}$ described in (41) and the received ciphertext $\overleftrightarrow{Cipher}\Big|^{m}$ to decrypt the message. This algorithm has two main parts: plaintext retrieval, which obtains the plaintext, and ciphertext verification, which checks if the ciphertext was altered after its creation. First, the message $\overleftrightarrow{M_1}\Big|^{m}$ is generated as shown in Equation (29). Then, by using H over $\overleftrightarrow{M_1}\Big|^{m}$, the array $\overleftrightarrow{H_{M_1}}\Big|^{m}$ is calculated. The original plaintext can be recovered from the first half of $\overleftrightarrow{M_1}\Big|^{m}$ as shown in Equation (30), and a pivot ciphertext $\overleftrightarrow{Cipher'}\Big|^{m}$ can be calculated. If the received ciphertext $\overleftrightarrow{Cipher}\Big|^{m}$ and the pivot ciphertext $\overleftrightarrow{Cipher'}\Big|^{m}$ match, then we know that the received ciphertext was not altered and the algorithm returns the deciphered plaintext. If $\overleftrightarrow{Cipher}\Big|^{m}$ and $\overleftrightarrow{Cipher'}\Big|^{m}$ do not match, then the received ciphertext $\overleftrightarrow{Cipher}\Big|^{m}$ was modified after its creation, so the algorithm returns the message *Invalid ciphertext received*.

C. O2MD2-III

This framework is the strongest variation of our system. The **O2MD2-III** framework not only provides ciphertext integrity verifications, but it also provides message authentication. The key generation and encryption algorithms have been designed in such a way, that the decryption algorithm can determine if a ciphertext was modified after its creation, and it also determines if a received ciphertext from an specific user was truly generated by him or not. This framework requires a public session of the form

$$\mathbb{S} = (m, \tilde{b}, r, H, \widetilde{H}, \tilde{h}, \tilde{k}, \tilde{s}), \tag{43}$$

where $m, \tilde{b}, r$ and H follow the same rules as in **O2MD2-II**, recalling that $2 \mid m$.
The positive integers $\tilde{h}, \tilde{k}$ and $\tilde{s}$ satisfy

$$\max(\tilde{s}[m\tilde{k} + 1], \tilde{s}[\frac{\tilde{h}m}{2} + 1]) < r, \tag{44}$$

and the one-way hash function $\widetilde{H}$ has the form

$$\widetilde{H} : \mathbb{Z}^m \to \left(\mathbb{Z}_{\tilde{h}}\right)^{\frac{m}{2}}. \tag{45}$$

1) KEY GENERATION

The key generation is shown in Algorithm 9. It takes the public session $\mathbb{S}$ described in (43), an instance $\overleftrightarrow{f}\Big|^{m}$ of f, a random positive integer $\tilde{a}$ and a random prime p_1 as input following inequality (33). This algorithm will generate four keys: two public keys K_{public} and K_σ, and two private keys K_{private} and K_s, as described in Equation (154). The new keys K_s and K_σ allow to perform message authentication, while the combination of these two and both K_{private} and K_{public} keep the message integrity check already provided by **O2MD2-II**. The key K_s consist of three $\frac{m}{2}$-sized arrays $\overleftrightarrow{f^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\Big|^{\frac{m}{2}}$, and $\overleftrightarrow{f^{\pm}}\Big|^{\frac{m}{2}}$, and the key K_σ consists of three arrays $\overleftrightarrow{K_{\sigma^+}}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{K_{\sigma^-}}\Big|^{\frac{m}{2}}$, and $\overleftrightarrow{K_\sigma}\Big|^{\frac{m}{2}}$, as described in (148) and (150), respectively. For two users A and B, if A wants to send a message to B, then he must encrypt the message using his signature private key K_{s_A} and B's public key K_{public_B}; when B receives the ciphertext, he must use his private key K_{private_B} and A's public signature key K_{σ_A} to decrypt it.

For two parties A and B, we will denote A's public and private keys as

$$\begin{aligned}
K_{\text{public}_A} &= (\overleftrightarrow{K}_{\text{public}_A}\Big|^{m}, p_2), \\
K_{\text{private}_A} &= (\overleftrightarrow{f}\Big|^{m}, p_1, \overleftrightarrow{F}_{p_1}\Big|^{m}, \overleftrightarrow{F}_{p_2}\Big|^{m}, \tilde{a}), \\
K_{\sigma_A} &= (\overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma_A^-}}\Big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma_A}}\Big|^{\frac{m}{2}}), \\
K_{s_A} &= (\overleftrightarrow{f^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^{\pm}}\Big|^{\frac{m}{2}}),
\end{aligned} \tag{46}$$

and B's keys as

$$\begin{aligned}
K_{\text{public}_B} &= (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m}, q_2), \\
K_{\text{private}_B} &= (\overleftrightarrow{g}\Big|^{m}, q_1, \overleftrightarrow{G}_{q_1}\Big|^{m}, \overleftrightarrow{G}_{q_2}\Big|^{m}, \tilde{\alpha}), \\
K_{\sigma_B} &= (\overleftrightarrow{K_{\sigma_B^+}}\Big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma_B^-}}\Big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma_B}}\Big|^{\frac{m}{2}}), \\
K_{s_B} &= (\overleftrightarrow{g^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{g^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{g^{\pm}}\Big|^{\frac{m}{2}}),
\end{aligned} \tag{47}$$

where p_1 and p_2 are the selected primes for A, and q_1 and q_2 the selected primes for B, with

$$\tilde{r} = \max(\tilde{b}, r) < q_1, \tag{48}$$

and by defining β as

$$\beta = \max(\overleftrightarrow{g}\big|^{m}), \tag{49}$$

we have that

$$q_2 > \max(q_1 m\tilde{a}\tilde{b}, m\beta\tilde{r}). \tag{50}$$

2) ENCRYPTION

The encryption algorithm for **O2MD2-III** is shown in Algorithm 10. It requires a session $\mathbb{S}$ as shown in (43), the sender's private signature key and receiver's public key to encrypt the message. The original message $\overleftrightarrow{M}\big|^{\frac{m}{2}}$ and a signature random salt $\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}$ are combined, and two different images $\overleftrightarrow{H_{M,\sigma}}\big|^{m}$ and $\overleftrightarrow{\widetilde{H}_{M,\sigma}}\big|^{\frac{m}{2}}$ under both one-way hash functions H and $\widetilde{H}$ are generated. Then $\overleftrightarrow{H_{M,\sigma}}\big|^{m}$ is used to generate a first sub-ciphertext, while the second image $\overleftrightarrow{\widetilde{H}_{M,\sigma}}\big|^{\frac{m}{2}}$ is used to generate three signature arrays $\overleftrightarrow{\sigma_M^{+}}\big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^{-}}\big|^{\frac{m}{2}}$ and $\overleftrightarrow{\sigma_M^{\pm}}\big|^{\frac{m}{2}}$. The one-way has function H is used over these three signature arrays to generate three additional sub-ciphertexts, and finally all sub-ciphertexts are combined to generate the final ciphertext.

3) DECRYPTION

The decryption algorithm for **O2MD2-III** is shown in Algorithm 11. It requires a session $\mathbb{S}$ as shown in (43), the receiver's private key and sender's public signature key to decrypt the message. The received ciphertext is split in four sub-ciphertexts and decrypted using the receiver's private key, generating four decrypted sub-messages $\overleftrightarrow{M_1}\big|^{m}$, $\overleftrightarrow{M_1^{+}}\big|^{m}$, $\overleftrightarrow{M_1^{-}}\big|^{m}$ and $\overleftrightarrow{M_1^{\pm}}\big|^{m}$. These messages are later used to check for the message authenticity and integrity. If both security checks pass, then a final message $\overleftrightarrow{M'}\big|^{\frac{m}{2}}$ equal to the original encrypted message is returned.

D. NOTES

Note that the **O2MD2-I** soft key-reset is embedded in the **O2MD2-I** key generation algorithm. If a user wants to refresh his keys, he can either run the soft key-reset and keep the same private key, or run the full key generation and generate a new pair of keys. Also, for all three frameworks, increasing the value of m by just some units will increase both the key space size and the cipherspace size exponentially. In the next section we present the proof of correctness for the decryption for the three frameworks.

V. PROOF OF CORRECTNESS

In this section we prove that the decryption algorithms for the three frameworks work correctly. Before doing so, we need to prove some additional minor theorems.

Theorem 1: Let $r, \tilde{a}, \tilde{b} \in \mathbb{Z}^{+}$ and primes p_1 and p_2 satisfy (33) and (35), and let $\overleftrightarrow{R}\big|^{m}_{(\tilde{a})}$ and $\overleftrightarrow{R}\big|^{m}_{(\tilde{b})}$ be two ephemeral arrays sampled from a discrete Gaussian distribution on $\mathbb{Z}$ and reduced modulo $\tilde{a}$ and $\tilde{b}$, respectively. Then they satisfy

$$\begin{aligned}&[p_1(\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})})](\bmod p_2)\\ &\qquad = p_1(\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})}),\end{aligned} \tag{51}$$

i.e.,

$$0 \le (p_1(\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})}))(i) < p_2, \tag{52}$$

for $i = 1, \ldots, m$.

Proof 1: For $\overleftrightarrow{R}\big|^{m}_{(\tilde{a})}$ and $\overleftrightarrow{R}\big|^{m}_{(\tilde{b})}$ sampled from a discrete Gaussian distribution on $\mathbb{Z}$ and reduced modulo $\tilde{a}$ and $\tilde{b}$, respectively, we know that they have the form $\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} = [r_1, \ldots, r_m]$ and $\overleftrightarrow{R}\big|^{m}_{(\tilde{b})} = [r'_1, \ldots, r'_m]$, with $0 \le r_i < \tilde{a}$ and $0 \le r'_i < \tilde{b}$, for $i = 1, \ldots, m$. Because all r_i and r'_i are non-negatives, the maximum value for any component of $\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})}$ is reached when $r_i = \tilde{a} - 1$ and $r'_i = \tilde{b} - 1$, for $i = 1, \ldots, m$. Then the maximum value of any component of the product is given by

$$\begin{aligned}&\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})}\\ &\quad = \underbrace{\Big[m(\tilde{a}-1)(\tilde{b}-1), \ldots, m(\tilde{a}-1)(\tilde{b}-1)\Big]}_{m\text{ times}},\end{aligned} \tag{53}$$

hence the maximum value for any component of $p_1(\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})})$ is $p_1 m(\tilde{a}-1)(\tilde{b}-1)$ and

$$p_1 m(\tilde{a}-1)(\tilde{b}-1) < p_1 m\tilde{a}\tilde{b}, \tag{54}$$

but we picked p_1 and p_2 satisfying (35), then any component of $p_1(\overleftrightarrow{R}\big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\big|^{m}_{(\tilde{b})})$ is between 0 and p_2, proving Equations (51) and (52). □

Theorem 2: For inputs $r, \tilde{a}, \tilde{b} \in \mathbb{Z}^{+}$ and for primes p_1 and p_2 following Equations (33) and (35), we have that

$$[\overleftrightarrow{f}\big|^{m} \circledast \overleftrightarrow{M}\big|^{m}](\bmod p_2) = \overleftrightarrow{f}\big|^{m} \circledast \overleftrightarrow{M}\big|^{m} \tag{55}$$

i.e.,

$$0 \le (\overleftrightarrow{f}\big|^{m} \circledast \overleftrightarrow{M}\big|^{m})(i) < p_2, \tag{56}$$

for $i = 1, \ldots, m$.

Proof 2: Similarly to the previous proof, each component of $\overleftrightarrow{f}\big|^{m}$ is a nonnegative integer bounded by b and each component of $\overleftrightarrow{M}\Big|^{m}$ is a nonnegative integer bounded by r, hence the maximum value of any component of $\overleftrightarrow{f}\Big|^{m} \circledast \overleftrightarrow{M}\Big|^{m}$ is mbr. Finally, because p_1 and p_2 follow the inequalities (33) and (35), then the values of all the components of $\overleftrightarrow{f}\Big|^{m} \circledast \overleftrightarrow{M}\Big|^{m}$ are between 0 and p_2, proving Equations (55) and (56). □

Theorem 3: The decryption algorithm for **O2MD2-I** generates a message $\overleftrightarrow{M_1}\Big|^{m}$ such that $\overleftrightarrow{M_1}\Big|^{m} = \overleftrightarrow{M}\Big|^{m}$.

Proof 3: From Equations (13), (16) and (18) in **O2MD2-I**, we get

$$\begin{aligned}\overleftrightarrow{K}_{\text{public}}\Big|^{m} &\equiv \text{Randomization}\left(\overleftrightarrow{F}_{p_2}\Big|^{m}, p_1, a\right)(\text{mod}\, p_2)\\ &\equiv p_1(\overleftrightarrow{F}_{p_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{a})})(\text{mod}\, p_2) \qquad (57)\end{aligned}$$

Using this in Equation (19), we get

$$\begin{aligned}\overleftrightarrow{R}\Big|^{m} &\equiv \overleftrightarrow{K}_{\text{public}}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})}(\text{mod}\, p_2)\\ &\equiv p_1(\overleftrightarrow{F}_{p_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})})(\text{mod}\, p_2) \qquad (58)\end{aligned}$$

and using this in Equation (20),

$$\begin{aligned}\overleftrightarrow{Cipher}\Big|^{m} \equiv\ &[p_1(\overleftrightarrow{F}_{p_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})})\\ &+ \overleftrightarrow{M}\Big|^{m}](\text{mod}\, p_2). \qquad (59)\end{aligned}$$

Using Equation (59) in $\overleftrightarrow{M_0}\Big|^{m}$ from (22),

$$\begin{aligned}\overleftrightarrow{M_0}\Big|^{m} &= [(\overleftrightarrow{Cipher}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m})(\text{mod}\, p_2)](\text{mod}\, p_1)\\ &\equiv [(p_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})}))(\text{mod}\, p_2)](\text{mod}\, p_1)\\ &\quad+[(\overleftrightarrow{M}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m})(\text{mod}\, p_2)](\text{mod}\, p_1). \qquad (60)\end{aligned}$$

By using Theorem 1 and Theorem 2 in the expression above,

$$\begin{aligned}\overleftrightarrow{M_0}\Big|^{m} \equiv\ &[\overleftrightarrow{M}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m}](\text{mod}\, p_1)\\ &+[p_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})})](\text{mod}\, p_1), \qquad (61)\end{aligned}$$

but $p_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})}) \equiv 0(\text{mod}\, p_1)$, so

$$\overleftrightarrow{M_0}\Big|^{m} \equiv [\overleftrightarrow{M}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m}](\text{mod}\, p_1). \qquad (62)$$

hence,

$$\overleftrightarrow{M_1}\Big|^{m} \equiv \overleftrightarrow{M_0}\Big|^{m} \circledast \overleftrightarrow{F}_{p_1}\Big|^{m} \equiv \overleftrightarrow{M}\Big|^{m}(\text{mod}\, p_1). \qquad (63)$$

Finally, p_1 satisfies (33), so $\overleftrightarrow{M}\Big|^{m}(\text{mod}\, p_1) = \overleftrightarrow{M}\Big|^{m}$, hence $\overleftrightarrow{M_1}\Big|^{m} = \overleftrightarrow{M}\Big|^{m}$. □

Theorem 4: The decryption algorithm for **O2MD2-II** generates a message $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$ such that $\overleftrightarrow{M'}\Big|^{\frac{m}{2}} = \overleftrightarrow{M}\Big|^{\frac{m}{2}}$.

Proof 4: Similarly to the previous proof, from Equation (13) and (25) in **O2MD2-II**, we get

$$\overleftrightarrow{K}_{\text{public}}\Big|^{m} \equiv p_1(\overleftrightarrow{F}_{p_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{a})})(\text{mod}\, p_2), \qquad (64)$$

hence

$$\begin{aligned}&\overleftrightarrow{K}_{\text{public}}\Big|^{m} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m}(\text{mod}\, p_2)\\ &\quad\equiv p_1(\overleftrightarrow{F}_{p_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m})(\text{mod}\, p_2), \qquad (65)\end{aligned}$$

and using (65) in (28), we get

$$\begin{aligned}\overleftrightarrow{Cipher}\Big|^{m} \equiv\ &[p_1(\overleftrightarrow{F}_{p_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m})\\ &+ \overleftrightarrow{M^*}\Big|^{m}](\text{mod}\, p_2). \qquad (66)\end{aligned}$$

And, from (29) in **O2MD2-II** decryption, we get

$$\begin{aligned}\overleftrightarrow{M_0}\Big|^{m} \equiv\ &[(p_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m}))(\text{mod}\, p_2)](\text{mod}\, p_1)\\ &+[(\overleftrightarrow{M^*}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m})(\text{mod}\, p_2)](\text{mod}\, p_1). \qquad (67)\end{aligned}$$

Now, from (27) we know that every component of the m-sized vector $\overleftrightarrow{H_{M^*}}\Big|^{m}$ is between 0 and $\tilde{b}$, and every component of $\overleftrightarrow{M^*}\Big|^{m}$ is between 0 and r. Therefore by using Theorem 1 and Theorem 2 in the expression above,

$$\begin{aligned}\overleftrightarrow{M_0}\Big|^{m} \equiv\ &[\overleftrightarrow{M^*}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m}](\text{mod}\, p_1)\\ &+[p_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m})](\text{mod}\, p_1), \qquad (68)\end{aligned}$$

but $p_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{a})} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{b})}) \equiv 0(\text{mod}\, p_1)$, so

$$\overleftrightarrow{M_0}\Big|^{m} \equiv [\overleftrightarrow{M^*}\Big|^{m} \circledast \overleftrightarrow{f}\Big|^{m}](\text{mod}\, p_1), \qquad (69)$$

hence

$$\overleftrightarrow{M_1}\Big|^{m} \equiv \overleftrightarrow{M_0}\Big|^{m} \circledast \overleftrightarrow{F}_{p_1}\Big|^{m} \equiv \overleftrightarrow{M^*}\Big|^{m}(\text{mod}\, p_1), \qquad (70)$$

but p_1 satisfies (33), so $\overleftrightarrow{M^*}\Big|^{m}(\text{mod}\, p_1) = \overleftrightarrow{M^*}\Big|^{m}$, and by (27),

$$\overleftrightarrow{M_1}\Big|^{m} = \overleftrightarrow{M^*}\Big|^{m} = \overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R}\Big|^{\frac{m}{2}}. \qquad (71)$$

Then, the value of $\overleftrightarrow{H_{M_1}}\Big|^{m}$ in (27) will be

$$\overleftrightarrow{H_{M_1}}\Big|^{m} = H(\overleftrightarrow{M_1}\Big|^{m}) = H(\overleftrightarrow{M^*}\Big|^{m}) = \overleftrightarrow{H_{M^*}}\Big|^{m}, \qquad (72)$$

so the value of $\overleftrightarrow{Cipher'}\Big|^{m}$ is

$$\begin{aligned}\overleftrightarrow{Cipher'}\Big|^{m} &= (\overleftrightarrow{K}_{\text{public}}\Big|^{m} \circledast \overleftrightarrow{H_{M_1}}\Big|^{m} + \overleftrightarrow{M_1}\Big|^{m})(\text{mod}\, p_2)\\ &= (\overleftrightarrow{K}_{\text{public}}\Big|^{m} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m} + \overleftrightarrow{M^*}\Big|^{m})(\text{mod}\, p_2)\\ &= \overleftrightarrow{Cipher}\Big|^{m}, \qquad (73)\end{aligned}$$

so the condition $\overleftrightarrow{Cipher'}\Big|^{m} == \overleftrightarrow{Cipher}\Big|^{m}$ holds, hence the **O2MD2-II**'s decryption must return $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$, and by (30), we know that

$$\overleftrightarrow{M'}\Big|^{\frac{m}{2}} = \overleftrightarrow{M_1}\Big|^{m}_{[1,\frac{m}{2}]} = \overleftrightarrow{M^*}\Big|^{m}_{[1,\frac{m}{2}]} = \overleftrightarrow{M}\Big|^{\frac{m}{2}}. \qquad (74)$$

□

Theorem 5: Let $\max(\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}})$ be the maximum value of any component of $\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}}$, and $\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}}$ in (155) and (158). Then

$$\max(\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}}) < r. \qquad (75)$$

Proof 5: Note that every component $\widehat{f}_i^+, \widehat{f}_i^-$, and $\widehat{f}_i^\pm$ of $\overleftrightarrow{f^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}}$, and every component $k_{\sigma_i}^+$, and $k_{\sigma_i}^-$ of $\overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{K_{\sigma_A^-}}\Big|^{\frac{m}{2}}$ in (155) satisfy $0 \le \widehat{f}_i^+, \widehat{f}_i^-, \widehat{f}_i^\pm < \tilde{s}$ and $0 \le k_{\sigma_i}^+, k_{\sigma_i}^- < \tilde{k}$, for $i = 1, \ldots, \frac{m}{2}$. Then

$$\max(\overleftrightarrow{f^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}}) = \frac{m}{2}(\tilde{s}-1)(\tilde{k}-1),$$
$$\max(\overleftrightarrow{f^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K_{\sigma_A^-}}\Big|^{\frac{m}{2}}) = \frac{m}{2}(\tilde{s}-1)(\tilde{k}-1),$$
$$\max(\overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}}) = \tilde{s}-1, \qquad (76)$$

so for $\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}$ in (155)

$$\begin{aligned}\max(\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}) &= 2[\frac{m}{2}(\tilde{s}-1)(\tilde{k}-1)] + (\tilde{s}-1) \\ &= (\tilde{s}-1)[m(\tilde{k}-1)+1] < \tilde{s}[m\tilde{k}+1],\end{aligned} \qquad (77)$$

and by (44), we know that

$$\max(\overleftrightarrow{\sigma}\big|^{\frac{m}{2}}) < \tilde{s}[m\tilde{k}+1] < r. \qquad (78)$$

Proceeding in a similar matter, every component f_i of $\overleftrightarrow{f^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}}$ in (148) satisfy $0 \le f_i < \tilde{s}$, for $i = 1, \ldots, \frac{m}{2}$. From (45), we also know that every component $\tilde{h}_i$ of $\overleftrightarrow{H_{M,\sigma}}\Big|^{\frac{m}{2}}$ satisfy $0 \le \tilde{h}_i < \tilde{h}$. Hence from (158) we know that

$$\max(\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}) = \max(\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}}) = \max(\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}}) \qquad (79)$$

and

$$\begin{aligned}\max(\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}) &= \frac{m}{2}(\tilde{s}-1)(\tilde{h}-1) + (\tilde{s}-1) \\ &= (\tilde{s}-1)[\frac{m}{2}(\tilde{h}-1)+1] < \tilde{s}[\frac{m\tilde{h}}{2}+1],\end{aligned} \qquad (80)$$

and by (44), we get

$$\max(\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}) < \tilde{s}[\frac{\tilde{h}m}{2}+1] < r. \qquad (81)$$

Combining (81) with (79), then we know that

$$\max(\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}) = \max(\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}}) = \max(\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}}) < r. \qquad (82)$$

Finally, from the expression above and (78), we deduct the inequality (75), proving the theorem. □

Theorem 6: The decryption algorithm for **O2MD2-III** generates a message $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$ such that $\overleftrightarrow{M'}\Big|^{\frac{m}{2}} = \overleftrightarrow{M}\Big|^{\frac{m}{2}}$.

Proof 6: Note that $\overleftrightarrow{C_0}\Big|^{m}$, $\overleftrightarrow{C_0^+}\Big|^{m}$, $\overleftrightarrow{C_0^-}\Big|^{m}$ and $\overleftrightarrow{C_0^\pm}\Big|^{m}$ in (161) are equal to $\overleftrightarrow{Cipher_0}\Big|^{m}$, $\overleftrightarrow{Cipher_1}\Big|^{m}$, $\overleftrightarrow{Cipher_2}\Big|^{m}$ and $\overleftrightarrow{Cipher_3}\Big|^{m}$ in (157) and (159). From (13) and (153) in **O2MD2-III**, using the notation in (47) for B, we get

$$\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \equiv q_1(\overleftrightarrow{G}_{q_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{\alpha})})(\text{mod } q_2). \qquad (83)$$

Define the set $\mathcal{M}$ as

$$\begin{aligned}\mathcal{M} = \{&\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^+}\Big|^{\frac{m}{2}}, \\ &\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}\}.\end{aligned} \qquad (84)$$

Let $\overleftrightarrow{M^*}\Big|^{m} \in \mathcal{M}$ and define $\overleftrightarrow{H_{M^*}}\Big|^{m}$ and $\overleftrightarrow{C_0^*}\Big|^{m}$ as

$$\overleftrightarrow{H_{M^*}}\Big|^{m} = H(\overleftrightarrow{M^*}\Big|^{m}), \qquad (85)$$

and

$$\overleftrightarrow{C_0^*}\Big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m} + \overleftrightarrow{M^*}\Big|^{m})(\text{mod } q_2). \qquad (86)$$

Then by using (83) in (86), we have

$$\begin{aligned}\overleftrightarrow{C_0^*}\Big|^{m} &= (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m} + \overleftrightarrow{M_*}\Big|^{m})(\text{mod } q_2) \\ &\equiv [q_1(\overleftrightarrow{G}_{q_2}\Big|^{m} \circledast \overleftrightarrow{R}\Big|^{m}_{(\tilde{\alpha})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m}) \\ &\quad + \overleftrightarrow{M^*}\Big|^{m}](\text{mod } q_2),\end{aligned} \qquad (87)$$

and for $\overleftrightarrow{M_0^*}\Big|^{m}$ defined as

$$\overleftrightarrow{M_0^*}\Big|^{m} = [(\overleftrightarrow{C_0^*}\Big|^{m} \circledast \overleftrightarrow{g}\big|^{m})(\text{mod } q_2)](\text{mod } q_1), \qquad (88)$$

we have

$$\begin{aligned}\overleftrightarrow{M_0^*}\Big|^{m} &= [(\overleftrightarrow{C_0^*}\Big|^{m} \circledast \overleftrightarrow{g}\Big|^{m})(\bmod q_2)](\bmod q_1)\\ &\equiv [(q_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{\alpha})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m}))(\bmod q_2)](\bmod q_1)\\ &\quad +[(\overleftrightarrow{M^*}\Big|^{m} \circledast \overleftrightarrow{g}\Big|^{m})(\bmod q_2)](\bmod q_1). \qquad (89)\end{aligned}$$

From **O2MD2-III** encryption, by definition, all the components of $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{R^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{R^+}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}$ are between 0 and r. Also, from Theorem 5, all the components of $\overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}}$, and $\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}}$ are also between 0 and r. Therefore, for all $\overleftrightarrow{M^*}\Big|^{m} \in \mathcal{M}$, all its components are between 0 and r. On the other hand, by (42), all the components of $\overleftrightarrow{H_{M^*}}\Big|^{m}$ in (85) are between 0 and $\tilde{b}$. Then by Theorem 1 and Theorem 2, we know

$$\begin{aligned}\overleftrightarrow{M_0^*}\Big|^{m} &\equiv [q_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{\alpha})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m})](\bmod q_1)\\ &\quad +[\overleftrightarrow{M^*}\Big|^{m} \circledast \overleftrightarrow{g}\Big|^{m}](\bmod q_1), \qquad (90)\end{aligned}$$

but $q_1(\overleftrightarrow{R}\Big|^{m}_{(\tilde{\alpha})} \circledast \overleftrightarrow{H_{M^*}}\Big|^{m}) \equiv 0(\bmod q_1)$, so

$$\overleftrightarrow{M_0^*}\Big|^{m} \equiv [\overleftrightarrow{M^*}\Big|^{m} \circledast \overleftrightarrow{g}\Big|^{m}](\bmod q_1), \qquad (91)$$

hence

$$\overleftrightarrow{M_1^*}\Big|^{m} \equiv \overleftrightarrow{M_0^*}\Big|^{m} \circledast \overleftrightarrow{G}_{q_1}\Big|^{m}(\bmod q_1) \equiv \overleftrightarrow{M^*}\Big|^{m}(\bmod q_1), \qquad (92)$$

and q_1 satisfies (48), so $\overleftrightarrow{M^*}\Big|^{m}(\bmod q_1) = \overleftrightarrow{M^*}\Big|^{m}$, therefore

$$\overleftrightarrow{M_1^*}\Big|^{m} = \overleftrightarrow{M^*}\Big|^{m}. \qquad (93)$$

Notice that, while $\overleftrightarrow{M^*}\Big|^{m}$ takes all the possible values $\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^-}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}$ in (84), then from (86), (157) and (161), we know that $\overleftrightarrow{C_0^*}\Big|^{m}$ takes all the possible values of $\overleftrightarrow{C_0}\Big|^{m}$, $\overleftrightarrow{C_0^+}\Big|^{m}$, $\overleftrightarrow{C_0^-}\Big|^{m}$ and $\overleftrightarrow{C_0^\pm}\Big|^{m}$, respectively. Also, from (88) and (162), x $\overleftrightarrow{M_0^*}\Big|^{m}$ takes all the values of $\overleftrightarrow{M_0}\Big|^{m}$, $\overleftrightarrow{M_0^+}\Big|^{m}$, $\overleftrightarrow{M_0^-}\Big|^{m}$ and $\overleftrightarrow{M_0^\pm}\Big|^{m}$ and from (92) and (163), $\overleftrightarrow{M_1^*}\Big|^{m}$ takes all the values of $\overleftrightarrow{M_1}\Big|^{m}$, $\overleftrightarrow{M_1^+}\Big|^{m}$, $\overleftrightarrow{M_1^-}\Big|^{m}$ and $\overleftrightarrow{M_1^\pm}\Big|^{m}$, respectively. Therefore, from (93):

$$\begin{aligned}\overleftrightarrow{M_1}\Big|^{m} &= \overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}; \quad \overleftrightarrow{M_1^+}\Big|^{m} = \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^+}\Big|^{\frac{m}{2}},\\ \overleftrightarrow{M_1^-}\Big|^{m} &= \overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^-}\Big|^{\frac{m}{2}}; \quad \overleftrightarrow{M_1^\pm}\Big|^{m} = \overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}},\end{aligned} \qquad (94)$$

and by using H on all the values of (94), from (164), we get

$$\begin{aligned}\overleftrightarrow{H_{M_1}}\Big|^{m} &= H(\overleftrightarrow{M_1}\Big|^{m}) = H(\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}),\\ \overleftrightarrow{H_{M_1^+}}\Big|^{m} &= H(\overleftrightarrow{M_1^+}\Big|^{m}) = H(\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^+}\Big|^{\frac{m}{2}}),\\ \overleftrightarrow{H_{M_1^-}}\Big|^{m} &= H(\overleftrightarrow{M_1^-}\Big|^{m}) = H(\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^-}\Big|^{\frac{m}{2}}),\\ \overleftrightarrow{H_{M_1^\pm}}\Big|^{m} &= H(\overleftrightarrow{M_1^\pm}\Big|^{m}) = H(\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}). \qquad (95)\end{aligned}$$

By using (94) and (95) on (167), for $\overleftrightarrow{C'}\Big|^{m}$ we get

$$\begin{aligned}\overleftrightarrow{C'}\Big|^{m} &= (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast \overleftrightarrow{H_{M_1}}\Big|^{m} + \overleftrightarrow{M_1}\Big|^{m})(\bmod q_2),\\ &= [\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast H(\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}})\\ &\quad + \overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}](\bmod q_2),\\ &= \overleftrightarrow{Cipher_0}\Big|^{m} = \overleftrightarrow{C_0}\Big|^{m}, \qquad (96)\end{aligned}$$

hence $\overleftrightarrow{C'}\Big|^{m} = \overleftrightarrow{C_0}\Big|^{m}$. In a similar matter, for $\overleftrightarrow{C_1^+}\Big|^{m}$, $\overleftrightarrow{C_1^-}\Big|^{m}$ and $\overleftrightarrow{C_1^\pm}\Big|^{m}$ defined in (167), we get $\overleftrightarrow{C_1^+}\Big|^{m} = \overleftrightarrow{C_0^+}\Big|^{m}$ and $\overleftrightarrow{C_1^-}\Big|^{m} = \overleftrightarrow{C_0^-}\Big|^{m}$ and $\overleftrightarrow{C_1^\pm}\Big|^{m} = \overleftrightarrow{C_0^\pm}\Big|^{m}$. Now, by using on (94) on (165) and (166), we get

$$\begin{aligned}\overleftrightarrow{M'}\Big|^{\frac{m}{2}} &= \overleftrightarrow{M_1}\Big|^{m}_{[1,\frac{m}{2}]} = \overleftrightarrow{M}\Big|^{\frac{m}{2}},\\ \overleftrightarrow{M_2^+}\Big|^{\frac{m}{2}} &= \overleftrightarrow{M_1^+}\Big|^{m}_{[1,\frac{m}{2}]} = \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}},\\ \overleftrightarrow{M_2^-}\Big|^{\frac{m}{2}} &= \overleftrightarrow{M_1^-}\Big|^{m}_{[1,\frac{m}{2}]} = \overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}},\\ \overleftrightarrow{M_2^\pm}\Big|^{\frac{m}{2}} &= \overleftrightarrow{M_1^\pm}\Big|^{m}_{[1,\frac{m}{2}]} = \overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}},\\ \overleftrightarrow{\sigma_0}\Big|^{\frac{m}{2}} &= \overleftrightarrow{M_1}\Big|^{m}_{[\frac{m}{2}+1,m]} = \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}. \qquad (97)\end{aligned}$$

Also, by (94) and (156),we get

$$\widetilde{H}(\overleftrightarrow{M_1}\Big|^{m}) = \widetilde{H}(\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}) = \overleftrightarrow{H_{M,\sigma}}\Big|^{\frac{m}{2}}. \qquad (98)$$

The expressions $\overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{M_2^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{K_{\sigma_A^-}}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{M_2^-}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{K_{\sigma_A}}\Big|^{\frac{m}{2}} \circledast \widetilde{H}(\overleftrightarrow{M_1}\Big|^{m})$ in (165) satisfy

$$\begin{aligned}&\overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{M_2^+}\Big|^{\frac{m}{2}}\\ &= \overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}}\end{aligned}$$

$$= \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast (\overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}})$$

$$= \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}}, \tag{99}$$

and

$$\overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{M}_2^-\Big|^{\frac{m}{2}}$$

$$= \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\sigma}_M^-\Big|^{\frac{m}{2}}$$

$$= \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast (\overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}})$$

$$= \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}}, \tag{100}$$

and

$$\overleftrightarrow{K}_{\sigma_A}\Big|^{\frac{m}{2}} \circledast \widetilde{H}(\overleftrightarrow{M}_1\Big|^{m})$$

$$= \overleftrightarrow{K}_{\sigma_A}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}$$

$$= \overleftrightarrow{f}^{\pm}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}$$
$$+ \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}. \tag{101}$$

By (99), (100), (101), and (158), for $\overleftrightarrow{\sigma'}\Big|^{\frac{m}{2}}$ defined in (165), we get

$$\overleftrightarrow{\sigma'}\Big|^{\frac{m}{2}} = \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{M}_2^+\Big|^{\frac{m}{2}} + \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{M}_2^-\Big|^{\frac{m}{2}}$$
$$+ \overleftrightarrow{M}_2^{\pm}\Big|^{\frac{m}{2}} - \overleftrightarrow{K}_{\sigma_A}\Big|^{\frac{m}{2}} \circledast \widetilde{H}(\overleftrightarrow{M}_1\Big|^{m})$$
$$= [\overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}$$
$$+ \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}}]$$
$$+[\overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}$$
$$+ \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}}]$$
$$+[\overleftrightarrow{f}^{\pm}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{f}^{\pm}\Big|^{\frac{m}{2}}]$$
$$-[\overleftrightarrow{f}^{\pm}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}} + \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}$$
$$+ \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}}_{M,\sigma}\Big|^{\frac{m}{2}}]$$

$$= \overleftrightarrow{K}_{\sigma_A^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{+}\Big|^{\frac{m}{2}} + \overleftrightarrow{K}_{\sigma_A^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{f}^{-}\Big|^{\frac{m}{2}} + \overleftrightarrow{f}^{\pm}\Big|^{\frac{m}{2}}$$
$$= \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}, \tag{102}$$

hence $\overleftrightarrow{\sigma'}\Big|^{\frac{m}{2}} = \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}$. Finally, by (166), the equality $\overleftrightarrow{\sigma'}\Big|^{\frac{m}{2}} = \overleftrightarrow{\sigma_0}\Big|^{\frac{m}{2}}$ holds, so **O2MD2-III** decryption returns $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$, and $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$ satisfies (97), therefore the decryption algorithm returns the original message $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$ as expected. $\square$

VI. SECURITY ANALYSIS

We perform a strong security analysis consisting of three different sections. First, we present the results of testing for indistinguishability over the generated ciphertexts against random noise. These tests are done by using the NIST *Statistical Test Suite*. Second, we provide twelve configurations to reach the security levels of AES-128, AES-192 and AES-256 (equivalent to the NIST *Post-Quantum Cryptography Standardization* project's security levels 1, 3 and 5, respectively). Third, we formally prove the message integrity and message authenticity verification capabilities of the **O2MD2-II** and **O2MD2-III** frameworks. Finally, we end this section with comments on additional types of attacks.

A. ANALYSIS FOR RANDOMNESS USING THE NIST STATISTICAL TEST SUITE

We used the NIST *Statistical Test Suite* (**Special Publication 800-22 Revision 1a**) to evaluate the indistinguishability of the generated ciphertexts against random noise [39], [41]. The NIST *Statistical Test Suite* takes a file as input, and performs a series of fifteen different tests on this input to check for various randomness-related properties. The input file must contain multiple lines of text already in binary form, and all these lines must have the same number of bits. The NIST *Statistical Test Suite*'s official documentation refers to these lines of text as *bit streams* and to the used number of bits as *length* [39]. The tests are then performed on each individual bit stream, and a final analysis report is given at the end. Examples of the performed tests are the *Frequency Test*, which calculates the proportion of zeros and ones, the *Frequency Within A Block Test*, which calculates the proportion of zeros and ones in blocks of different bit sizes, the *Runs Test*, which calculates the total number of uninterrupted sequences of identical bits, and others [40].

Our experiment was implemented on Ubuntu 20.04 LTS using the official NIST *Statistical Test Suite* repository and it was built with gcc 7.0.5 [41]. We followed the *ECRYPT Benchmarking of Cryptographic Systems (eBACS)* interface proposed by the *Virtual Application and Implementation Research Lab (VAMPIRE)* to make a *c* program that generates 100 ciphertexts for 100 random plaintexts [20], [21]. These 100 ciphertexts are then converted into binary form and stacked in a single file, which is finally fed to the NIST *Statistical Test Suite* software.

TABLE 4. Results of our experiment using the NIST *Statistical Test Suite* over 100 different ciphertexts to test for indistinguishability against random noise (most of the *NonOverlappingTemplate* results were trimmed for display purposes). A statistical test is considered to be passed if at least 96 out of the 100 bit streams pass it.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
12	16	9	11	4	8	10	10	9	11	0.494392	98/100	Frequency
10	14	8	12	9	11	11	12	8	5	0.739918	100/100	BlockFrequency
10	11	13	8	7	10	6	12	8	15	0.616305	98/100	CumulativeSums
12	10	12	11	5	12	7	13	5	13	0.437274	98/100	CumulativeSums
14	3	12	9	13	9	8	6	12	14	0.213309	98/100	Runs
11	9	7	11	13	9	8	9	10	13	0.935716	100/100	LongestRun
18	0	35	0	0	0	47	0	0	0	0.000000	100/100	Rank
10	12	7	7	8	24	0	10	12	10	0.000157	98/100	FFT
7	5	17	0	21	0	0	0	50	0	0.000000	100/100	OverlappingTemplate
49	27	13	4	1	1	3	2	0	0	0.000000	96/100	ApproximateEntropy
11	3	3	14	20	8	18	6	7	10	0.000320	100/100	Serial
7	11	5	19	13	4	10	10	8	13	0.042808	100/100	Serial

Based on the NIST *Statistical Test Suite* final analysis report, the input file passes a given test if at least 96 out of the 100 bit streams pass that specific test. A summary of the output report obtained after our experiment is shown in Table 4. From this table, we know that our ciphertexts pass the following tests: **(1)** The *Frequency Test*, **(2)** The *Frequency Within A Block Test*, **(3)** The *Cumulative Sum Test*, **(4)** The *Runs Test*, **(5)** The *Longest Run of Ones In A Block Test*, **(6)** The *Random Binary Matrix Rank Test*, **(7)** The *Discrete Fourier Transform Test*, **(8)** The *Overlapping Template Matching Test*, **(9)** The *Approximate Entropy Test*, and **(10)** The *Serial Test*. Brief descriptions of these ten tests are shown in Table 5, and more detailed explanations are provided in the NIST *Statistical Test Suite*'s official documentation [39], [40].

B. COMPARISON AGAINST THE OTHER NIST CANDIDATES

The NIST *Post-Quantum Cryptography Standardization* project officially proposes five different security levels in its *Security Evaluation Criteria* section, and they are sorted in an increasing order based on their strength [42]. Security level 1 attacks, security level 3 attacks and security level 5 attacks must be equivalent to perform key searches on block ciphers with 128-bit keys, 192-bit keys and 256-bit keys, respectively. Security level 2 attacks and security level 4 attacks must be equivalent to perform collision searches on 256-bit and 384-bit hash functions, respectively. In other words, security level 1, security level 3 and security level 5 compliant cryptosystems must be as hard to break as AES-128, AES-192 and AES-256, respectively. Similarly, security level 2 and security level 4 compliant cryptosystems must be as hard to break as SHA256/SHA3-256 and SHA384/SHA3-384 [42].

We compared our framework against seven popular candidates on the second round of the NIST *Post-Quantum Cryptography Standardization* project. These cryptosystems are NTRU, NTRU Prime, FrodoKEM, Crystals-Kyber, ThreeBears, NewHope and LAC [14], [23]–[26], [31], [32]. These seven cryptosystems proposed a total of twenty-six different implementations. The proposed four implementations for NTRU are ntru-hps2048509, ntru-hps2048677, ntru-hrs701, and ntru-hps4096821 [31]. The six proposed ones for NTRU Prime are ntrulpr653, sntrup653, ntrulpr761, sntrup761, ntrulpr857, and sntrup857 [32]. The three proposed ones for FrodoKEM are FrodoKEM-640, FrodoKEM-976, and FrodoKEM-1344, while the three proposals for Crystals-Kyber are kyber512, kyber768 and kyber1024 [25], [26]. The three implementations proposed for ThreeBears are BabyBear, MamaBear and PapaBear [14], while the four proposed ones for NewHope are NewHope512cca, NewHope512cpa, NewHope1024cca, and NewHope1024cpa [24]. Finally, the three implementations proposed for LAC are LAC-KEM-128, LAC-KEM-195 and LAC-KEM-256 [23]. Table 6 shows how all these twenty-six implementations cover the five security levels for the NIST *Post-Quantum Cryptography Standardization* project.

We propose different configurations that reach the NIST *Post-Quantum Cryptography Standardization* project's security levels 1, 3 and 5. Because we want to implement our framework in high-end devices and low-level hardware, our configurations also have different memory requirements between them. We consider as low-level hardware to all the devices that support a 32-bit datatype. For example, the Arduino and Raspberry pi devices support the *unsigned long* datatype which stores 32 bits (4 bytes) [43]. Similarly, we consider as high-end devices to all the devices that support a 64-bit datatype. For example, personal computers with 64-bit CPUs support the 64 bits (8 bytes) *unsigned long long* datatype.

Our framework's key generation algorithm depends on the session variable $\mathbb{S}$ (described in Equations (32), (41) and (43) for **O2MD2-I**, **O2MD2-II** and **O2MD2-III**, respectively), the values of $\tilde{a}$, b, p_1 and p_2 (for **O2MD2-I**, **O2MD2-II** and **O2MD2-III**), and $\tilde{h}$ $\tilde{k}$ and $\tilde{s}$ (for **O2MD2-III**). As different security levels can be attained by modifying all these parameters, we provide six different configurations based on their memory sizes. We classify as *Low Security*, *Middle Security* and *High Security* to our configurations capable of reaching the NIST *Post-Quantum Cryptography Standardization* project's security levels 1, 3 and 5, respectively. Our configurations' names follow the notation code **OXYZAB**, where **O** stands for O2MD^2, **XYZ** may have $L32$ or $H64$ as their value, depending if it is designed for

TABLE 5. Brief descriptions of the ten tests of the NIST *Statistical Test Suite* that were successfully passed by the generated ciphertexts in our experiment. More detailed descriptions are provided in the NIST *Statistical Test Suite*'s official documentation [39], [40].

No.	Test Name	Focus	Purpose
1	Frequency Test	The proportion of zeros and ones for the entire sequence.	Determines if the number of zeros and ones in a sequence are approximately the same as what is expected in a truly random sequence.
2	Frequency Within A Block Test	The proportion of zeros and ones within M-bit blocks.	Determines if the frequency of ones is an M-bit block is approximately $M/2$.
3	Cumulative Sum (Cusum) Test	The maximal excursion (from zero) of the random walk defined by the cumulative sum in the sequence.	Determines if the cumulative sum of the partial sequences occurring in the tested sequence is different to the expected behavior of that cumulative sum for random sequences.
4	Runs Test	The total number of zeros and ones runs in the entire sequence.	Determines if the number of runs of consecutive ones and zeros of various lengths is as expected as for a random sequence.
5	Longest Run Of Ones In A Block Test	The longest run of ones within M-bit blocks.	Determines if the length of the longest run of ones is consistent with the length of the longest run of ones expected in a random sequence.
6	Random Binary Matrix Rank Test	The rank of disjoint sub-matrices of the entire sequence.	Checks for linear dependence among fixed length substrings of the original sequence.
7	Discrete Fourier Transform Test	The peak heights in the discrete Fast Fourier Transform.	Detects repetitive patterns in the tested sequence that would indicate a deviation from true randomness.
8	Overlapping Template Matching Test	The number of pre-defined target substrings.	Rejects sequences that show deviations from the expected number of runs of ones of a given length.
9	Approximate Entropy Test	The frequency of each and every overlapping M-bit pattern.	Compares the frequency of overlapping blocks of two consecutive lengths (M and $M+1$) against what is expected for a random sequence
10	Serial Test	The frequency of each and every overlapping M-bit pattern across the entire sequence.	Determines if the number of occurrences of the $2M$ M-bit overlapping patterns is approximately the same as the expected for a random sequence.

TABLE 6. Classification of the twenty-six implementations of the NIST *Post-Quantum Cryptography Standardization* project candidates in terms of their security levels. The twelve proposed implementations for our framework are also shown.

Cryptosystem	NIST Security Level				
	1	2	3	4	5
NTRU	ntru-hps2048509	N/A	ntru-hps2048677 ntru-hrs701	N/A	ntru-hps4096821
NTRU Prime	N/A	ntrulpr653 sntrup653	ntrulpr761 sntrup761	ntrulpr857 sntrup857	N/A
Crystal	kyber512	N/A	kyber768	N/A	kyber1024
Frodo	FrodoKEM-640	N/A	FrodoKEM-976	N/A	FrodoKEM-1344
ThreeBears	N/A	BabyBear	N/A	MamaBear	PapaBear
NewHope	NewHope512cca NewHope512cpa	N/A	N/A	N/A	NewHope1024cca NewHope1024cpa
LAC	LAC-KEM-128	N/A	LAC-KEM-192	N/A	LAC-KEM-256
O2MD2	OL3216 OH6416 OL3216FFT OH6416FFT	N/A	OL3232 OH6432 OL3232FFT OH6432FFT	N/A	OL3264 OH6464 OL3264FFT OH6464FFT

a low-level hardware (capable of storing 32-bit variables) or high-end devices (capable of storing 64-bit variables), and **AB** may be 16, 32 or 64, depending on the desired size for *m*.

For low-level hardware, we propose the configurations OL3216, OL3232 and OL3264, while for high-end devices we propose the configurations OH6416, OH6432 and OH6464. Besides these six implementations, we also propose their Fast Fourier Transform (FFT) versions (which use the Number Theoretic Transform NTT). As with Crystals-Kyber, NewHope and LAC, the Number Theoretic Transform could be used to perform polynomial multiplications over a ring. We propose then the OL3216FFT, OL3232FFT and OL3264FFT configurations for low-level hardware, and OH6416FFT, OH6432FFT and OH6464FFT high-end devices. One important requirement for a device to implement the FFT version is that it needs to support floating point operations, as complex roots of unity are needed to perform the NTT products.

Finally, all twelve proposed configurations are classified into different security levels. For the Low Security Level we have OL3216, OH6416 and their FFT versions OL3216FFT and OH6416FFT. For the Middle Security Level we have OL3232, OH6432 and their FFT versions OL3232FFT and OH6432FFT. For the High Security Level we have OL3264, OH6464 and their FFT versions OL3264FFT and OH6464FFT. All these twelve configurations' security levels and memory requirements are shown in Table 7. Finally,

TABLE 7. Twelve different configurations of the O2MD^2 cryptosystem. These configurations are categorized by their security level and usage of Fast Fourier Transform (FFT) when performing the multiplication operation. The necessary bit sizes for all the variables are also displayed.

No	Security Levels	Implementation	Uses FFT	Parameters									
				m	$\tilde{b}$	r	$\tilde{a}$	b	p_1	p_2	$\tilde{h}$	$\tilde{k}$	$\tilde{s}$
1	Low Security	OL3216	No	16	16 bits	16 bits	16 bits	16 bits	16 bits	32 bits	N/A	N/A	N/A
2		OH6416		16	32 bits	32 bits	32 bits	32 bits	32 bits	64 bits	N/A	N/A	N/A
3		OL3216FFT	Yes	16	16 bits	16 bits	16 bits	16 bits	16 bits	32 bits	N/A	N/A	N/A
4		OH6416FFT		16	32 bits	32 bits	32 bits	32 bits	32 bits	64 bits	N/A	N/A	N/A
5	Middle Security	OL3232	No	32	16 bits	16 bits	16 bits	16 bits	16 bits	32 bits	N/A	N/A	N/A
6		OH6432		32	32 bits	64 bits	32 bits	32 bits	64 bits	64 bits	16 bits	16 bits	16 bits
7		OL3232FFT	Yes	32	16 bits	16 bits	16 bits	16 bits	16 bits	32 bits	N/A	N/A	N/A
8		OH6432FFT		32	32 bits	64 bits	32 bits	32 bits	64 bits	64 bits	16 bits	16 bits	16 bits
9	High Security	OL3264	No	64	16 bits	16 bits	16 bits	16 bits	16 bits	32 bits	N/A	N/A	N/A
10		OH6464		64	32 bits	64 bits	32 bits	32 bits	64 bits	64 bits	16 bits	16 bits	16 bits
11		OL3264FFT	Yes	64	16 bits	16 bits	16 bits	16 bits	16 bits	32 bits	N/A	N/A	N/A
12		OH6464FFT		64	32 bits	64 bits	32 bits	32 bits	64 bits	64 bits	16 bits	16 bits	16 bits

Figure 2 shows the security levels of the non-FFT configurations against AES-128, AES-192 and AES-256.

C. ABOUT MESSAGE INTEGRITY AND MESSAGE AUTHENTICITY VERIFICATIONS

The **O2MD2-II** and **O2MD2-III** frameworks only decrypt valid ciphertexts generated from valid messages. In the decryption algorithms, several integrity verification clauses are checked for different parts of the ciphertexts prior to return a plaintext, and if one of these verifications fails, an error message is returned. Below we formally prove that the probability of randomly guessing a valid ciphertext for both frameworks becomes negligible as m increases. Also, we prove that the probability of randomly guessing a valid ciphertext that satisfies the authenticity verification check in the **O2MD2-III** framework is also negligible as m increases.

Theorem 7: The **O2MD2-II** and **O2MD2-III** frameworks only decrypt valid ciphertexts that pass the integrity check, and an adversary cannot randomly generate a valid ciphertext, except with negligible probability.

Proof 7: For the **O2MD2-II** framework, let $\overleftrightarrow{Cipher}\Big|^{m}$ be a received ciphertext. From **O2MD2-II** decryption, we know that a message $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$ is returned only if $\overleftrightarrow{Cipher}\Big|^{m}$ passes the validation of $\overleftrightarrow{Cipher}\Big|^{m} == \overleftrightarrow{Cipher'}\Big|^{m}$, where $\overleftrightarrow{Cipher'}\Big|^{m}$ is defined in (31), and $0 \leq \overleftrightarrow{M'}\Big|^{\frac{m}{2}}(i) < r$, for $i = 1, \ldots, m$. Hence, $\overleftrightarrow{Cipher}\Big|^{m}$ must be a valid ciphertext of a valid message. Now, denote as $\mathbf{Guess_{II}}$ to the event of randomly guessing a valid ciphertext from all the possible ciphertext space. Then $\Pr[\mathbf{Guess_{II}}]$ is uper bounded by

$$\Pr[\mathbf{Guess_{II}}] \leq (\tfrac{r}{p_2})^m, \tag{103}$$

but from (35) and (33) we know that

$$p_2 > \max(p_1 m\tilde{a}\tilde{b}, mb\tilde{r}) > rm\max(\tilde{a}\tilde{b}, b), \tag{104}$$

and by taking $\tilde{a} \geq 2$, $\tilde{b} \geq 2$ and $b \geq 2$, we know that

$$p_2 > rm\max(\tilde{a}\tilde{b}, b) > rm\max(4, 2) > 2rm, \tag{105}$$

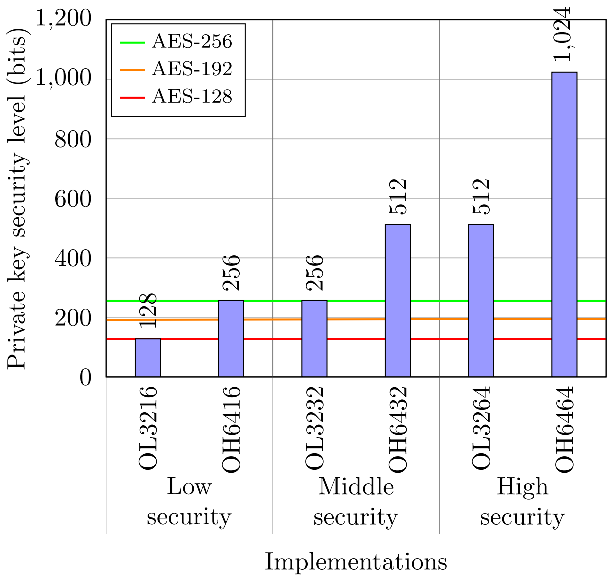


FIGURE 2. Six of the twelve proposed implementations of the O2MD^2 system. These are: (1) Low Security: OL3216 and OH6416. (2) Middle Security: OL3232 and OH6432. (3) High Security: OL3264 and OH6464. The FFT versions are omitted. The Low Security level, Middle Security Level and High Security Level comply with the NIST *Post-Quantum Cryptography Standardization* project's security levels 1, 3 and 5, respectively. The benchmarks for AES-128, AES-192 and AES-256 are also shown for clarity purposes.

so

$$\frac{1}{2rm} > \frac{1}{p_2}, \tag{106}$$

hence

$$\Pr[\mathbf{Guess_{II}}] \leq (\tfrac{r}{p_2})^m < (\tfrac{r}{2rm})^m = \tfrac{1}{(2m)^m}, \tag{107}$$

so for any $\varepsilon > 0$, we can always find a value of m_ε of m for which $\Pr[\mathbf{Guess_{II}}] < \varepsilon$ holds for all $m > m_\varepsilon$.

Similarly for the **O2MD2-III** framework, let $\overleftrightarrow{C}\Big|^{4m}$ be a received ciphertext. From **O2MD2-III** decryption, we know that if a message $\overleftrightarrow{M'}\Big|^{\frac{m}{2}}$ is returned, then the conditions ϕ_1

and ϕ_2 defined as

$$\phi_1 = (\overleftrightarrow{C'}\Big|^m == \overleftrightarrow{C_0}\Big|^m) \wedge (\overleftrightarrow{C_1^+}\Big|^m == \overleftrightarrow{C_0^+}\Big|^m) \wedge (\overleftrightarrow{C_1^-}\Big|^m == \overleftrightarrow{C_0^-}\Big|^m) \wedge (\overleftrightarrow{C_1^\pm}\Big|^m == \overleftrightarrow{C_0^\pm}\Big|^m), \tag{108}$$

$$\phi_2 = \bigwedge_{i=1}^{m}[(0 \le \overleftrightarrow{M_1}\Big|^m (i) < r) \wedge (0 \le \overleftrightarrow{M_1^+}\Big|^m (i) < r) \wedge (0 \le \overleftrightarrow{M_1^-}\Big|^m (i) < r) \wedge (0 \le \overleftrightarrow{M_1^\pm}\Big|^m (i) < r)], \tag{109}$$

are both true. From (109) we know that $\overleftrightarrow{M_1}\Big|^m$, $\overleftrightarrow{M_1^+}\Big|^m$, $\overleftrightarrow{M_1^-}\Big|^m$ and $\overleftrightarrow{M_1^\pm}\Big|^m$ are valid messages, and by (167), the calculated $\overleftrightarrow{C'}\Big|^m$, $\overleftrightarrow{C_1^+}\Big|^m$, $\overleftrightarrow{C_1^-}\Big|^m$ and $\overleftrightarrow{C_1^\pm}\Big|^m$ are valid ciphertexts from valid messages. Then by (108), we know that $\overleftrightarrow{C_0}\Big|^m$, $\overleftrightarrow{C_0^+}\Big|^m$, $\overleftrightarrow{C_0^-}\Big|^m$ and $\overleftrightarrow{C_0^\pm}\Big|^m$ are also valid ciphertexts of valid messages. But they are the four different parts of the received $\overleftrightarrow{C}\Big|^{4m}$, as shown in (161). Therefore, the received $\overleftrightarrow{C}\Big|^{4m}$ must be a valid ciphertext of a valid original message $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$. Now, denote as $\mathbf{Guess_{III}}$ to the event of randomly guessing a valid ciphertext $\overleftrightarrow{C}\Big|^{4m}$ for the **O2MD2-III** framework. Note that $\overleftrightarrow{C}\Big|^{4m}$ is a valid ciphertext if and only if its four parts $\overleftrightarrow{C_0}\Big|^m$, $\overleftrightarrow{C_0^+}\Big|^m$, $\overleftrightarrow{C_0^-}\Big|^m$ and $\overleftrightarrow{C_0^\pm}\Big|^m$ are also valid ciphertexts built from the **O2MD2-III** encryption. By denoting as $\mathbf{Guess_{III}} \ge N$ to the event of randomly guessing at least N valid parts out of the four, we have

$$\Pr[\mathbf{Guess_{III}}] < \Pr[\mathbf{Guess_{III}} \ge 1] = 1 - \Pr[\mathbf{Guess_{III}} = 0]. \tag{110}$$

For each one of the four parts of a valid ciphertext, there are r^m valid sub-ciphertexts out of p_2^m possible ones, hence

$$\Pr[\mathbf{Guess_{III}} = 0] = (\frac{p_2^m - r^m}{p_2^m})^4 = (1 + [-\frac{r^m}{p_2^m}])^4. \tag{111}$$

Let

$$x = -\frac{r^m}{p_2^m}. \tag{112}$$

From (33) and (35), we know that $r < p_2$, so $\frac{r}{p_2} < 1$, hence $-\frac{r}{p_2} > -1$. Therefore $x > -1$, and by using the Bernoulli's inequality over (111), we get

$$\Pr[\mathbf{Guess_{III}} = 0] = (1 + [-\frac{r^m}{p_2^m}])^4 \ge 1 + 4[-\frac{r^m}{p_2^m}], \tag{113}$$

then

$$-\Pr[\mathbf{Guess_{III}} = 0] \le -1 + \frac{4r^m}{p_2^m}. \tag{114}$$

By using (114) with (110), we get

$$\Pr[\mathbf{Guess_{III}}] < 1 - \Pr[\mathbf{Guess_{III}} = 0] < 1 - 1 + \frac{4r^m}{p_2^m} = \frac{4r^m}{p_2^m}, \tag{115}$$

and from (106), we get a higher upper bound

$$\Pr[\mathbf{Guess_{III}}] < \frac{4r^m}{p_2^m} < 4(\frac{r}{2rm})^m = \frac{4}{(2m)^m}, \tag{116}$$

so for any $\varepsilon > 0$, we can always find a value of m_ε of m for which $\Pr[\mathbf{Guess_{III}}] < \varepsilon$ holds for all $m > m_\varepsilon$.

Therefore both **O2MD2-II** and **O2MD2-III** frameworks only decrypt valid ciphertexts that pass the integrity check, and an adversary cannot randomly generate a valid ciphertext, except with negligible probability. □

Theorem 8: The **O2MD2-III** framework only decrypts valid ciphertexts that satisfy the authenticity check performed over the sender's public signature key, and an adversary cannot randomly generate such ciphertexts, except with negligible probability.

Proof 8: From (99), (100), (101) and (102) in the proof of correctness of **O2MD2-III** framework, we know that the decryption algorithm only decrypts ciphertexts that satisfy the authenticity check performed over the sender's public signature key. For an adversary to randomly guess such a ciphertext, he must correctly guess all $\overleftrightarrow{M_2^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{M_2^-}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{M_2^\pm}\Big|^{\frac{m}{2}}$ and $\widetilde{H}(\overleftrightarrow{M_1}\Big|^m)$ simultaneously. Denote as $\mathbf{Guess^*_{III}}$ to the event of correctly guessing these four messages. Then we know that

$$\Pr[\mathbf{Guess^*_{III}}] < (\frac{1}{r^{\frac{m}{2}}})^3 \times (\frac{1}{\tilde{h}^{\frac{m}{2}}}) = (\frac{1}{r^3\tilde{h}})^{\frac{m}{2}}, \tag{117}$$

so for any $\varepsilon > 0$, we can always find a value of m_ε of m for which $\Pr[\mathbf{Guess^*_{III}}] < \varepsilon$ holds for all $m > m_\varepsilon$. □

Theorem 9: The **O2MD2-III** framework only decrypts valid ciphertexts that satisfy both the integrity and authenticity checks, and an adversary cannot randomly generate such ciphertexts, except with negligible probability.

Proof 9: Let $\overleftrightarrow{C}\Big|^{4m}$ be a ciphertext that the **O2MD2-III** framework can decipher correctly. Then by Theorem 7, it must pass the integrity check. At the same time, by Theorem 8, it must also pass the authenticity check. Therefore, $\overleftrightarrow{C}\Big|^{4m}$ must pass both verifications simultaneously. Denote as $\mathbf{Guess'_{III}}$ to the event of correctly guessing a ciphertext that pass both checks. Then

$$\Pr[\mathbf{Guess'_{III}}] \le \Pr[\mathbf{Guess_{III}}] \times \Pr[\mathbf{Guess^*_{III}}], \tag{118}$$

by using (116) and (117) on (118), we get

$$\Pr[\mathbf{Guess'_{III}}] \le \Pr[\mathbf{Guess_{III}}] \times \Pr[\mathbf{Guess^*_{III}}] < (\frac{4}{(2m)^m})(\frac{1}{r^3\tilde{h}})^{\frac{m}{2}} = \frac{4}{(4m^2r^3\tilde{h})^{\frac{m}{2}}}, \tag{119}$$

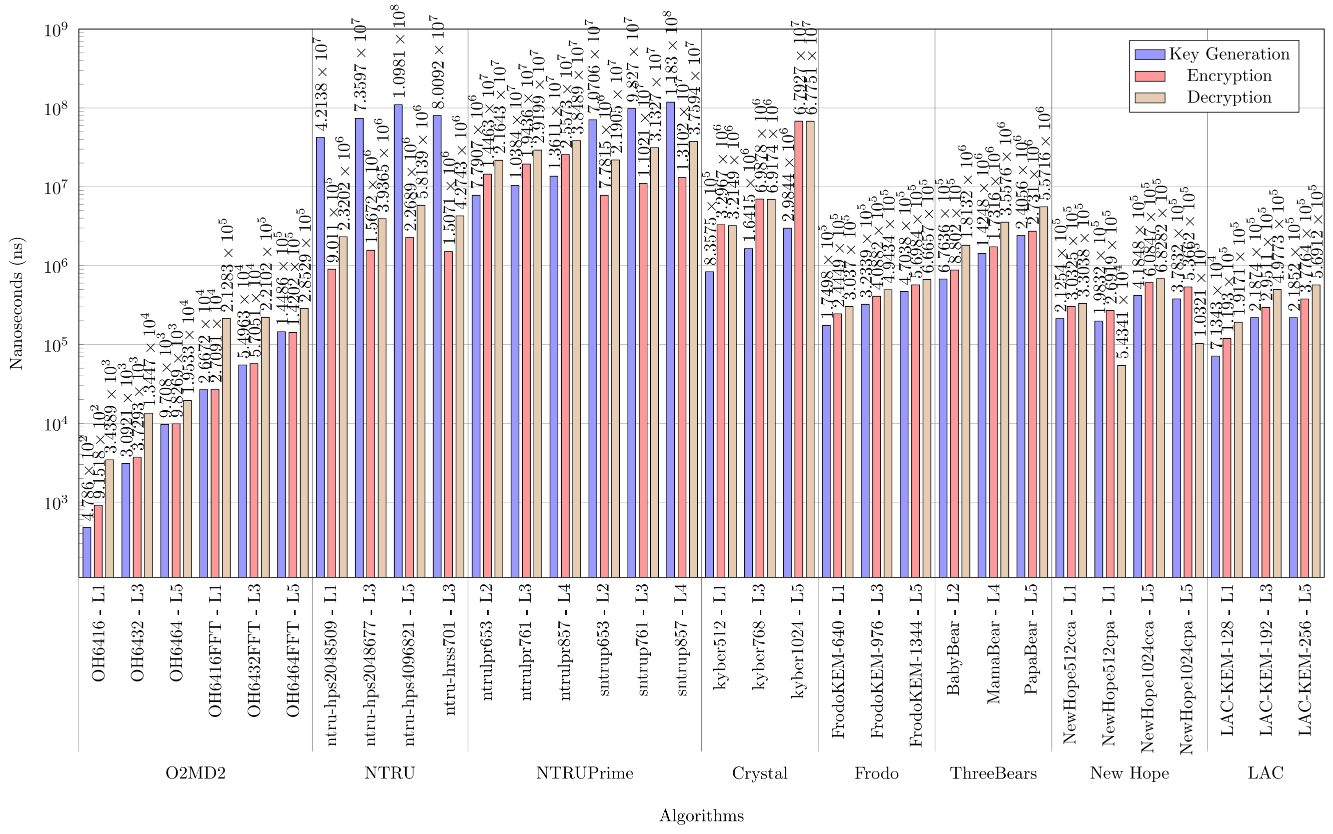


FIGURE 3. Speed comparison of the six configurations proposed for high-end devices (OH6416, OH6432, OH6464, OH6416FFT, OH6432FFT, and OH6464FFT using the O2MD2-I framework), and the twenty-six implementations for NTRU, NTRU Prime, Crystals-Kyber, FrodoKEM, ThreeBears, NewHope and LAC (making a total of thirty-two implementations). One hundred different cycles consisting of key generation, encryption and decryption were made for each implementation, and the average execution time for each algorithm is shown in nanoseconds (logarithmic scale is used in this diagram for the ordinate axis). The reached NIST security level for each implementation is also shown.

so for any $\varepsilon > 0$, we can always find a value of m_ε of m for which $\Pr[\mathbf{Guess}'_{\mathbf{III}}] < \varepsilon$ holds for all $m > m_\varepsilon$. □

D. FINAL NOTES ABOUT THE SECURITY ANALYSIS

As the **O2MD2-II** and **O2MD2-III** frameworks require to use one-way hash functions, the security of the message integrity and message authenticity verifications will depend on how strong the selected one-way hash functions are. In an already implemented system, if a vulnerability of the used one-way hash functions is discovered, a re-implementation of the system is suggested.

Also, we use the rings $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$ while encrypting and decrypting. These types of rings have been widely studied and are considered secure, but similarly to the authors of ThreeBears, NTRU and NTRU Prime, we also claim that it is difficult to verify all possible weaknesses, and it is unknown if vulnerabilities over the ring learning with errors (RLWE) problem exist that have not been discovered yet.

Similar to what was done by the ThreeBears authors, our Middle Security Level and High Security Level configurations exceed the requirements proposed by the NIST *Post-Quantum Cryptography Standardization* project for security levels 3 and 5. This was done as a failsafe in case that any new quantum-based attack is discovered in the future which forces to re-design all the post-quantum candidates.

Finally, linear differential attacks have also been considered. Since we use both rings $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$ (and add random salts in the **O2MD2-II** and **O2MD2-III** encryption algorithms), for big m, p_1 and p_2, any linear property between the plaintext and ciphertext is lost, making it difficult to perform any kind of attacks based on small differences in plaintexts. Differential attacks require as much work as brute force attacks, and with the correct set up of the initial parameters and a periodical key refresh, the message space can be impossible to scan entirely.

VII. PERFORMANCE ANALYSIS

We created a testing environment to test our configurations (non-FFT and FFT versions) for high-end devices against the twenty-six official implementations provided by NTRU, NTRU Prime, FrodoKEM, Crystals-Kyber, ThreeBears, HewHope and LAC in the NIST *Post-Quantum Cryptography Standardization* project website [19]. We used Ubuntu 20.04 on a machine with an Intel i7-7700HQ CPU and 32 Gb of RAM.

Our implementations followed the *ECRYPT Benchmarking of Cryptographic Systems (eBACS)* interface. Tests of 100 cycles were performed. Each cycle consists of a key generation, encryption of a randomly generated plaintext, decryption of the resulting ciphertext and comparison of the original plaintext and deciphered message. The NIST *Post-Quantum Cryptography Standardization* project candidates used plaintexts of 48 bytes, while our implementations used plaintexts of 64 bytes. The execution times of the key generation, encryption and decryption of all the thirty-two tested algorithms were measured in nanoseconds. Figure 3 shows the average result of a side-by-side speed test of our implementations of OH6416, OH6432, OH6464, OH6416FFT, OH6432FFT, and OH6464FFT (using the **O2MD2-I** framework) against all the other twenty-six implementations (logarithmic scale is used for the ordinate axis). On average, our implementations perform their key generation, encryption and decryption from 2 to 4 orders of magnitude faster than the twenty-six tested NIST *Post-Quantum Cryptography Standardization* project candidates' implementations.

Finally, we must mention that our framework, unlike the tested candidates, can be implemented using only unsigned integers. We use the rings $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$, so all of the operations always use non-negative values. This creates a significant advantage in terms of memory management, as bit packing can be performed easier and less memory is needed to reach a desired security level compared to the tested candidates. All the code for our experiment is published in our GitHub's official repository [44].

VIII. CONCLUSION AND FUTURE WORK

The O2MD^2 cryptosystem is an innovative post-quantum cryptosystem that uses the concept of arithmetic functions to construct a one-to-many private/public key architecture. It provides a distributed key refresh to all the users in a network. It is based in the ring learning with errors (RLWE) problem, defined over the rings $\mathbb{Z}_{p_1}[x]/\langle x^m - 1\rangle$ and $\mathbb{Z}_{p_2}[x]/\langle x^m - 1\rangle$.

Our cryptosystem has three different frameworks: **O2MD2-I**, **O2MD2-II** and **O2MD2-III**, and each one of them provides useful capabilities. The **O2MD2-I** framework has a distributed and backwards-compatible key refresh, while the **O2MD2-II** and **O2MD2-III** frameworks provide message integrity verifications, which are useful to identify modified ciphertexts. Additionally, the **O2MD2-III** framework also performs message authenticity verifications, so a receiver can determine if a message was created by the sender or not.

Our solution also creates ciphertexts that are indistinguishable from random noise. By use of the NIST *Statistical Test Suite* over 100 ciphertexts of 100 randomly generated plaintexts, we prove that the generated ciphertexts have ten fundamental properties of random noise.

We provide twelve suggested configurations that can reach the NIST *Post-Quantum Cryptography Standardization* project's security levels 1, 3 and 5. These configurations are also classified by their memory requirements, where six of them are designed to run in high-end devices, while the other six on low-level hardware. After testing against twenty-six different implementations of seven popular cryptosystems from the *Post-Quantum Cryptography Standardization* project Round 2, we checked that our implementations perform their key generation, encryption and decryption from 2 to 4 orders of magnitude faster than the tested candidates.

Based on the algebraic properties we used throughout this work, we have observed that the O2MD^2 cryptosystem offers several interesting characteristics. The one-to-many private/public key approach, the distributed key refresh, the use of arithmetic functions as private keys, the message integrity and authenticity verifications, the indistinguishability of the ciphertexts against random noise, and the flexibility to reach different security levels and fast performance make this proposal a valuable contribution for the cryptographic community.

For future work, we will address several topics. First, we will fully explore the IND-CPA and IND-CCA1/IND-CCA2 game scenarios. The **O2MD2-II** and **O2MD2-III** frameworks were designed in such a way that, when analyzing the IND-CPA game scenario, the Fujisaki-Okamoto transformation could be used to reach IND-CCA security in the random oracle model. Also, we will analyze lattice-based attacks (for example, LLL-based attacks). Second, we will create hardware implementations using FPGAs and ASICs. Third, we will develop communication and handshake protocols considering the key refresh in a network of users with high latency. Furthermore, we plan to explore the use of O2MD^2 on the Internet of Things (IoT) world, as the key generation can be modified to work as a block chain process. Finally, we will explore on how to create a chain-of-trust in a network of users while retaining secure communication.

APPENDIX
EXAMPLE

This is an example using **O2MD2-I**. Define the session as $\mathbb{S} = (m, \tilde{b}, r) = (5, 120, 120)$. Let f be a secret arithmetic function. Assume that, for a secret instance I, the device generated $\overleftrightarrow{f}\Big|^5 = [2, 81, 27, 9, 3]$.

A. KEY GENERATION

All the entries of $\overleftrightarrow{f}\Big|^5$ are non-negatives. Select $p_1 = 251$ and $\tilde{a} = 120$. Note that

$$\begin{aligned} b &= \max(\overleftrightarrow{f}\Big|^5) = 81, \\ \tilde{r} &= \max(\tilde{b}, r) = 120, \end{aligned} \tag{120}$$

Let p_2 be a random prime satisfying inequality (35), i.e.,

$$p_2 > \max(p_1 m \tilde{a} \tilde{b}, m b \tilde{r}) = \max(18072000, 48600) \tag{121}$$

The prime 18072001 satisfies[1] the inequality (121). From (8) we know that

$$\overleftrightarrow{F}_{251}\Big|^{5} = [92, 223, 74, 164, 128],$$

$$\overleftrightarrow{F}_{18072001}\Big|^{5} = [11798464, 16030112, 7407741, 1287507, 11026277]. \quad (122)$$

Our private key is

$$K_{\text{private}} = \left(\overleftrightarrow{f}\Big|^{5}, 251, \overleftrightarrow{F}_{251}\Big|^{5}, \overleftrightarrow{F}_{18072001}\Big|^{5}, 120\right), \quad (123)$$

and we have to run the soft key-reset algorithm to generate our public key.

B. SOFT KEY-RESET

Note that $\tilde{a} = 120$. By sampling five different integers from a discrete Gaussian distribution over $\mathbb{Z}$ and reducing them modulo $\tilde{a}$, we perform a randomization over $\overleftrightarrow{F}_{18072001}\Big|^{5}$ with the array

$$\overleftrightarrow{R}\Big|^{5}_{(120)} = [98, 83, 38, 114, 4], \quad (124)$$

getting

$$\overleftrightarrow{K}_{\text{public}}\Big|^{5} = [5728821, 15683333, 5171087, 12284834, 13126654]. \quad (125)$$

We can perform a secondary randomization over $\overleftrightarrow{F}_{18072001}\Big|^{5}$ with a new array

$$\overleftrightarrow{R^{*}}\Big|^{5}_{(120)} = [58, 53, 77, 85, 90], \quad (126)$$

generating an alternative public key

$$\overleftrightarrow{K^{*}}_{\text{public}}\Big|^{5} = [12818350, 12426167, 13811533, 10953056, 17687579]. \quad (127)$$

Finally, set K_{public} and K^{*}_{public} as

$$K_{\text{public}} = \left(\overleftrightarrow{K}_{\text{public}}\Big|^{5}, 18072001\right),$$

$$K^{*}_{\text{public}} = \left(\overleftrightarrow{K^{*}}_{\text{public}}\Big|^{5}, 18072001\right). \quad (128)$$

We will use both public keys in the encryption algorithm to show that the **O2MD2-I** one-to-many private/public key architecture works correctly.

[1] Any prime greater than 18072001 will also be a valid candidate prime for p_2.

C. ENCRYPTION

We encrypt the ASCII code of the message **Hello**, obtaining

$$\overleftrightarrow{M}\Big|^{5} = [72, 101, 108, 108, 111]. \quad (129)$$

Note that $\tilde{b} = 120$. By sampling five different integers from a discrete Gaussian distribution over $\mathbb{Z}$ and reducing them modulo $\tilde{b}$, we make a randomization over $\overleftrightarrow{K}_{\text{public}}\Big|^{5}$ from (121) with the array

$$\overleftrightarrow{R_{encrypt_1}}\Big|^{5}_{(120)} = [52, 45, 91, 95, 22], \quad (130)$$

getting

$$\overleftrightarrow{R}\Big|^{5} = \text{Randomization}\left(\overleftrightarrow{K}_{\text{public}}\Big|^{5}, 1, 120\right)$$
$$= [3321923152, 2842804607, 3548678919, 3013267698, 3131717969], \quad (131)$$

then the ciphertext is given by

$$\overleftrightarrow{Cipher}\Big|^{5} = \left(\overleftrightarrow{M}\Big|^{5} + \overleftrightarrow{R}\Big|^{5}\right)(\text{mod } 18072001)$$
$$13315640, 5261907]. \quad (132)$$

We also make a new randomization again over $\overleftrightarrow{K}_{\text{public}}\Big|^{5}$ from (125) but with a second random vector

$$\overleftrightarrow{R_{encrypt_2}}\Big|^{5}_{(120)} = [17, 23, 45, 90, 2], \quad (133)$$

getting

$$\overleftrightarrow{R_1}\Big|^{5} = \text{Randomization}\left(\overleftrightarrow{K}_{\text{public}}\Big|^{5}, 1, 120\right)$$
$$= [2161360827, 1448885025, 2105056208, 1912390611, 1575374362], \quad (134)$$

where we can generate the alternative ciphertext

$$\overleftrightarrow{Cipher_1}\Big|^{5} = \left(\overleftrightarrow{M}\Big|^{5} + \overleftrightarrow{R_1}\Big|^{5}\right)(\text{mod } 18072001)$$
$$= [10792780, 3125046, 8704200, 14830614, 3110386]. \quad (135)$$

By using the alternative second public key $\overleftrightarrow{K^{*}}_{\text{public}}\Big|^{5}$ from (127), if we use the random vectors

$$\overleftrightarrow{R_{encrypt_3}}\Big|^{5}_{(120)} = [33, 81, 78, 19, 14],$$

$$\overleftrightarrow{R_{encrypt_4}}\Big|^{5}_{(120)} = [13, 25, 19, 92, 54], \quad (136)$$

we generate two new alternative ciphertexts:

$$\overleftrightarrow{Cipher^{*}}\Big|^{5} = [18005199, 1895209, 12634479, 5802146, 12936752],$$

$$\overleftrightarrow{Cipher^{*}_1}\Big|^{5} = [17286247, 11666092, 5342822, 6738991, 2816645]. \quad (137)$$

Algorithm 9 (O2MD2-III) Key generation.

```
input : Instance $\overleftrightarrow{f}\big|^{m}$, session $\mathbb{S} = (m, \tilde{b}, r, H, \widetilde{H}, \tilde{k}, \tilde{s})$ with $2 \mid m$, random prime $p_1$ with $\tilde{r} = \max(\tilde{b}, r) < p_1$, and
        random positive integer $\tilde{a}$.
output: Message: *Cannot construct keys based on the inputs*, or the set of keys
        $K_{\text{public}} = (\overleftrightarrow{K}_{\text{public}}\big|^{m}, p_2);\; K_{\text{private}} = (\overleftrightarrow{f}\big|^{m}, p_1, \overleftrightarrow{F}_{p_1}\big|^{m}, \overleftrightarrow{F}_{p_2}\big|^{m}, \tilde{a}),$
        $K_{\sigma} = (\overleftrightarrow{K_{\sigma^+}}\big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma^-}}\big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma}}\big|^{\frac{m}{2}});\; K_s = (\overleftrightarrow{f^+}\big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\big|^{\frac{m}{2}}, \overleftrightarrow{f^{\pm}}\big|^{\frac{m}{2}}).$

1  if $\overleftrightarrow{f}\big|^{m}$ *contains at least one negative component* then
2      Return *Cannot construct keys based on the inputs.*
3  end
4  Let $\overleftrightarrow{f^+}\big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\big|^{\frac{m}{2}}$, and $\overleftrightarrow{f^{\pm}}\big|^{\frac{m}{2}}$ be three $\frac{m}{2}$-sized vectors where each element is sampled from a discrete Gaussian
   distribution on $\mathbb{Z}$ and reduced modulo $\tilde{s}$.
5  Define $K_s$ as
       $K_s = (\overleftrightarrow{f^+}\big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\big|^{\frac{m}{2}}, \overleftrightarrow{f^{\pm}}\big|^{\frac{m}{2}}).$          (148)
6  Let $\overleftrightarrow{K_{\sigma^+}}\big|^{\frac{m}{2}}$ and $\overleftrightarrow{K_{\sigma^-}}\big|^{\frac{m}{2}}$ be two $\frac{m}{2}$-sized vectors where each element is sampled from a discrete Gaussian distribution on $\mathbb{Z}$
   and reduced modulo $\tilde{k}$.
7  Calculate
       $\overleftrightarrow{K_{\sigma}}\big|^{\frac{m}{2}} = \overleftrightarrow{f^+}\big|^{\frac{m}{2}} \circledast \overleftrightarrow{K_{\sigma^+}}\big|^{\frac{m}{2}} + \overleftrightarrow{f^-}\big|^{\frac{m}{2}} \circledast \overleftrightarrow{K_{\sigma^-}}\big|^{\frac{m}{2}} + \overleftrightarrow{f^{\pm}}\big|^{\frac{m}{2}}.$          (149)
8  Define $K_{\sigma}$ as
       $K_{\sigma} = (\overleftrightarrow{K_{\sigma^+}}\big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma^-}}\big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma}}\big|^{\frac{m}{2}})$          (150)
9  Calculate
       $b = \max(\overleftrightarrow{f}\big|^{m}),$
       $\tilde{r} = \max(\tilde{b}, r).$          (151)
10 Select a random prime $p_2$ such that
       $p_2 > \max(p_1 m \tilde{a} \tilde{b}, m b \tilde{r}).$          (152)
11 if $\overleftrightarrow{F}_{p_1}\big|^{m}$ *or* $\overleftrightarrow{F}_{p_2}\big|^{m}$ *do not exist* then
12     Select a different prime $p_1$ or $p_2$, respectively. Restart Key Generation.
13 end
14 Calculate
       $\overleftrightarrow{K}_{\text{public}}\big|^{m} = \text{Randomization}\left(\overleftrightarrow{F}_{p_2}\big|^{m}, p_1, \tilde{a}\right) \pmod{p_2}.$          (153)
15 Return
       $K_{\text{public}} = (\overleftrightarrow{K}_{\text{public}}\big|^{m}, p_2);\; K_{\text{private}} = (\overleftrightarrow{f}\big|^{m}, p_1, \tilde{a}),$
       $K_{\sigma} = (\overleftrightarrow{K_{\sigma^+}}\big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma^-}}\big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma}}\big|^{\frac{m}{2}});\; K_s = (\overleftrightarrow{f^+}\big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\big|^{\frac{m}{2}}, \overleftrightarrow{f^{\pm}}\big|^{\frac{m}{2}}).$          (154)
```

Algorithm 10 (O2MD2-III) Encryption. An entity A sends a message to an entity B.

```
input  : Receiver's public key $K_{\text{public}_B} = (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m}, q_2)$, session $\mathbb{S} = (m, \tilde{b}, r, H, \widetilde{H}, \tilde{k}, \tilde{s})$, sender's public signature
         key $K_{\sigma_A} = (\overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma_A^-}}\Big|^{\frac{m}{2}}, \overleftrightarrow{K_{\sigma_A}}\Big|^{\frac{m}{2}})$, sender's private signature key $K_{s_A} = (\overleftrightarrow{f^+}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^-}\Big|^{\frac{m}{2}}, \overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}})$, and
         Message $\overleftrightarrow{M}\Big|^{\frac{m}{2}}$.
output : Ciphertext $\overleftrightarrow{Cipher}\Big|^{4m}$.
1 Let $\overleftrightarrow{f^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{f^-}\Big|^{\frac{m}{2}}$, and $\overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}}$ be three $\frac{m}{2}$-sized vectors where each element is sampled from a discrete Gaussian
  distribution on $\mathbb{Z}$ and reduced modulo $\tilde{s}$.
2 Calculate
      $\overleftrightarrow{\sigma}\Big|^{\frac{m}{2}} = \overleftrightarrow{f^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K_{\sigma_A^+}}\Big|^{\frac{m}{2}} + \overleftrightarrow{f^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{K_{\sigma_A^-}}\Big|^{\frac{m}{2}} + \overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}}.$   (155)
3 Calculate
      $\overleftrightarrow{H_{M,\sigma}}\Big|^{m} = H(\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}),$
      $\overleftrightarrow{\widetilde{H}_{M,\sigma}}\Big|^{\frac{m}{2}} = \widetilde{H}(\overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}).$   (156)
4 Calculate
      $\overleftrightarrow{Cipher_0}\Big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast \overleftrightarrow{H_{M,\sigma}}\Big|^{m} + \overleftrightarrow{M}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{\sigma}\Big|^{\frac{m}{2}}) (\text{mod } q_2).$   (157)
5 Calculate
      $\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} = \overleftrightarrow{f^+}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}_{M,\sigma}}\Big|^{\frac{m}{2}} + \overleftrightarrow{f^+}\Big|^{\frac{m}{2}},$
      $\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} = \overleftrightarrow{f^-}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}_{M,\sigma}}\Big|^{\frac{m}{2}} + \overleftrightarrow{f^-}\Big|^{\frac{m}{2}},$   (158)
      $\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} = \overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}} \circledast \overleftrightarrow{\widetilde{H}_{M,\sigma}}\Big|^{\frac{m}{2}} + \overleftrightarrow{f^\pm}\Big|^{\frac{m}{2}}.$
6 Let $\overleftrightarrow{R^+}\Big|^{\frac{m}{2}}$, $\overleftrightarrow{R^-}\Big|^{\frac{m}{2}}$ and $\overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}$ be three $\frac{m}{2}$-sized vectors where each element is sampled from a discrete Gaussian
  distribution on $\mathbb{Z}$ and reduced modulo $r$.
7 Calculate
      $\overleftrightarrow{Cipher_1}\Big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast H(\overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^+}\Big|^{\frac{m}{2}}) + \overleftrightarrow{\sigma_M^+}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^+}\Big|^{\frac{m}{2}}) (\text{mod } q_2),$
      $\overleftrightarrow{Cipher_2}\Big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast H(\overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^-}\Big|^{\frac{m}{2}}) + \overleftrightarrow{\sigma_M^-}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^-}\Big|^{\frac{m}{2}}) (\text{mod } q_2),$   (159)
      $\overleftrightarrow{Cipher_3}\Big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\Big|^{m} \circledast H(\overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}) + \overleftrightarrow{\sigma_M^\pm}\Big|^{\frac{m}{2}} \oplus \overleftrightarrow{R^\pm}\Big|^{\frac{m}{2}}) (\text{mod } q_2).$
8 Return
      $\overleftrightarrow{Cipher}\Big|^{4m} = \overleftrightarrow{Cipher_0}\Big|^{m} \oplus \overleftrightarrow{Cipher_1}\Big|^{m} \oplus \overleftrightarrow{Cipher_2}\Big|^{m} \oplus \overleftrightarrow{Cipher_3}\Big|^{m}.$   (160)
```

Algorithm 11 (O2MD2-III) Decryption. An entity B decrypts a message submitted by an entity A

```
input : Ciphertext $\overleftrightarrow{C}\big|^{4m}$, session
        $\mathbb{S} = (m, \tilde{b}, r, H, \widetilde{H}, \tilde{k}, \tilde{\mathfrak{s}})$, receiver's public key
        $K_{\text{public}_B} = (\overleftrightarrow{K}_{\text{public}_B}\big|^{m}, q_2)$, receiver's private key
        $K_{\text{private}_B} = (\overleftrightarrow{g}\big|^{m}, q_1, \tilde{\alpha})$, and sender's public
        signature key $K_{\sigma_A} = (\overleftrightarrow{K}_{\sigma_A^+}\big|^{\frac{m}{2}}, \overleftrightarrow{K}_{\sigma_A^-}\big|^{\frac{m}{2}}, \overleftrightarrow{K}_{\sigma_A}\big|^{\frac{m}{2}})$.
output: Message $\overleftrightarrow{M'}\big|^{\frac{m}{2}}$.
1  Define $\overleftrightarrow{C_0}\big|^{m}$, $\overleftrightarrow{C_0^+}\big|^{m}$, $\overleftrightarrow{C_0^-}\big|^{m}$ and $\overleftrightarrow{C_0^\pm}\big|^{m}$ as
       $\overleftrightarrow{C_0}\big|^{m} = \overleftrightarrow{C}\big|^{4m}_{[1,m]}$; $\overleftrightarrow{C_0^+}\big|^{m} = \overleftrightarrow{C}\big|^{4m}_{[m+1,2m]}$,
       $\overleftrightarrow{C_0^-}\big|^{m} = \overleftrightarrow{C}\big|^{4m}_{[2m+1,3m]}$; $\overleftrightarrow{C_0^\pm}\big|^{m} = \overleftrightarrow{C}\big|^{4m}_{[3m+1,4m]}$.      (161)
2  Calculate $\overleftrightarrow{M_0}\big|^{m}$, $\overleftrightarrow{M_0^+}\big|^{m}$, $\overleftrightarrow{M_0^-}\big|^{m}$ and $\overleftrightarrow{M_0^\pm}\big|^{m}$ as
       $\overleftrightarrow{M_0}\big|^{m} = [(\overleftrightarrow{C_0}\big|^{m} \circledast \overleftrightarrow{g}\big|^{m})(\text{mod } q_2)](\text{mod } q_1)$,
       $\overleftrightarrow{M_0^+}\big|^{m} = [(\overleftrightarrow{C_0^+}\big|^{m} \circledast \overleftrightarrow{g}\big|^{m})(\text{mod } q_2)](\text{mod } q_1)$,
       $\overleftrightarrow{M_0^-}\big|^{m} = [(\overleftrightarrow{C_0^-}\big|^{m} \circledast \overleftrightarrow{g}\big|^{m})(\text{mod } q_2)](\text{mod } q_1)$,
       $\overleftrightarrow{M_0^\pm}\big|^{m} = [(\overleftrightarrow{C_0^\pm}\big|^{m} \circledast \overleftrightarrow{g}\big|^{m})(\text{mod } q_2)](\text{mod } q_1)$.      (162)
3  Calculate $\overleftrightarrow{M_1}\big|^{m}$, $\overleftrightarrow{M_1^+}\big|^{m}$, $\overleftrightarrow{M_1^-}\big|^{m}$ and $\overleftrightarrow{M_1^\pm}\big|^{m}$ as
       $\overleftrightarrow{M_1}\big|^{m} = \overleftrightarrow{M_0}\big|^{m} \circledast \overleftrightarrow{G}_{q_1}\big|^{m} (\text{mod } q_1)$,
       $\overleftrightarrow{M_1^+}\big|^{m} = \overleftrightarrow{M_0^+}\big|^{m} \circledast \overleftrightarrow{G}_{q_1}\big|^{m} (\text{mod } q_1)$,
       $\overleftrightarrow{M_1^-}\big|^{m} = \overleftrightarrow{M_0^-}\big|^{m} \circledast \overleftrightarrow{G}_{q_1}\big|^{m} (\text{mod } q_1)$,
       $\overleftrightarrow{M_1^\pm}\big|^{m} = \overleftrightarrow{M_0^\pm}\big|^{m} \circledast \overleftrightarrow{G}_{q_1}\big|^{m} (\text{mod } q_1)$.      (163)
4  Calculate $\overleftrightarrow{H_{M_1}}\big|^{m}$, $\overleftrightarrow{H_{M_1^+}}\big|^{m}$, $\overleftrightarrow{H_{M_1^-}}\big|^{m}$ and $\overleftrightarrow{H_{M_1^\pm}}\big|^{m}$ as
       $\overleftrightarrow{H_{M_1}}\big|^{m} = H(\overleftrightarrow{M_1}\big|^{m})$; $\overleftrightarrow{H_{M_1^+}}\big|^{m} = H(\overleftrightarrow{M_1^+}\big|^{m})$,
       $\overleftrightarrow{H_{M_1^-}}\big|^{m} = H(\overleftrightarrow{M_1^-}\big|^{m})$; $\overleftrightarrow{H_{M_1^\pm}}\big|^{m} = H(\overleftrightarrow{M_1^\pm}\big|^{m})$.      (164)
5  Define $\overleftrightarrow{M'}\big|^{\frac{m}{2}}$, $\overleftrightarrow{M_2^+}\big|^{\frac{m}{2}}$, $\overleftrightarrow{M_2^-}\big|^{\frac{m}{2}}$ and $\overleftrightarrow{M_2^\pm}\big|^{\frac{m}{2}}$ as
       $\overleftrightarrow{M'}\big|^{\frac{m}{2}} = \overleftrightarrow{M_1}\big|^{m}_{[1,\frac{m}{2}]}$; $\overleftrightarrow{M_2^+}\big|^{\frac{m}{2}} = \overleftrightarrow{M_1^+}\big|^{m}_{[1,\frac{m}{2}]}$,
       $\overleftrightarrow{M_2^-}\big|^{\frac{m}{2}} = \overleftrightarrow{M_1^-}\big|^{m}_{[1,\frac{m}{2}]}$; $\overleftrightarrow{M_2^\pm}\big|^{\frac{m}{2}} = \overleftrightarrow{M_1^\pm}\big|^{m}_{[1,\frac{m}{2}]}$.      (165)
6  Define $\overleftrightarrow{\sigma_0}\big|^{\frac{m}{2}}$ as
       $\overleftrightarrow{\sigma_0}\big|^{\frac{m}{2}} = \overleftrightarrow{M_1}\big|^{m}_{[\frac{m}{2}+1,m]}$      (166)
7  Calculate $\overleftrightarrow{C'}\big|^{m}$, $\overleftrightarrow{C_1^+}\big|^{m}$, $\overleftrightarrow{C_1^-}\big|^{m}$ and $\overleftrightarrow{C_1^\pm}\big|^{m}$ as
       $\overleftrightarrow{C'}\big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\big|^{m} \circledast \overleftrightarrow{H_{M_1}}\big|^{m} + \overleftrightarrow{M_1}\big|^{m})(\text{mod } q_2)$,
       $\overleftrightarrow{C_1^+}\big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\big|^{m} \circledast \overleftrightarrow{H_{M_1^+}}\big|^{m} + \overleftrightarrow{M_1^+}\big|^{m})(\text{mod } q_2)$,
       $\overleftrightarrow{C_1^-}\big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\big|^{m} \circledast \overleftrightarrow{H_{M_1^-}}\big|^{m} + \overleftrightarrow{M_1^-}\big|^{m})(\text{mod } q_2)$,
       $\overleftrightarrow{C_1^\pm}\big|^{m} = (\overleftrightarrow{K}_{\text{public}_B}\big|^{m} \circledast \overleftrightarrow{H_{M_1^\pm}}\big|^{m} + \overleftrightarrow{M_1^\pm}\big|^{m})(\text{mod } q_2)$.      (167)
8  if $\overleftrightarrow{C'}\big|^{m} \neq \overleftrightarrow{C_0}\big|^{m}$ Or $\overleftrightarrow{C_1^+}\big|^{m} \neq \overleftrightarrow{C_0^+}\big|^{m}$ Or $\overleftrightarrow{C_1^-}\big|^{m} \neq \overleftrightarrow{C_0^-}\big|^{m}$
   Or $\overleftrightarrow{C_1^\pm}\big|^{m} \neq \overleftrightarrow{C_0^\pm}\big|^{m}$ Or $\overleftrightarrow{M_1}\big|^{m}(i) \geq r$ Or $\overleftrightarrow{M_1^+}\big|^{m}(i) \geq r$ Or
   $\overleftrightarrow{M_1^-}\big|^{m}(i) \geq r$ Or $\overleftrightarrow{M_1^\pm}\big|^{m}(i) \geq r$, for some
   $i = 1, \ldots, m$ then
9      Return Invalid ciphertext received.
10 end
11 Calculate $\overleftrightarrow{\sigma'}\big|^{\frac{m}{2}}$ as
       $\overleftrightarrow{\sigma'}\big|^{\frac{m}{2}} = \overleftrightarrow{K}_{\sigma_A^+}\big|^{\frac{m}{2}} \circledast \overleftrightarrow{M_2^+}\big|^{\frac{m}{2}} + \overleftrightarrow{K}_{\sigma_A^-}\big|^{\frac{m}{2}} \circledast \overleftrightarrow{M_2^-}\big|^{\frac{m}{2}}$
           $+ \overleftrightarrow{M_2^\pm}\big|^{\frac{m}{2}} - \overleftrightarrow{K}_{\sigma_A}\big|^{\frac{m}{2}} \circledast \widetilde{H}(\overleftrightarrow{M_1}\big|^{m})$      (168)
12 if $\overleftrightarrow{\sigma'}\big|^{\frac{m}{2}} == \overleftrightarrow{\sigma_0}\big|^{\frac{m}{2}}$ then
13     Return $\overleftrightarrow{M'}\big|^{\frac{m}{2}}$.
14 else
15     Return Message Authenticity failed.
16 end
```

D. DECRYPTION

For the ciphertext in (132), we have that

$$\begin{aligned}&(\overleftrightarrow{Cipher}\Big|^{5} \circledast \overleftrightarrow{f}\Big|^{5})(\bmod 18072001)\\ &\quad = [4741098, 5305912, 5220083, 4408431, 6184511],\end{aligned} \tag{138}$$

then

$$\begin{aligned}\overleftrightarrow{M_0}\Big|^{5} &= [4741098, 5305912, 5220083, 4408431,\\ &\quad 6184511](\bmod 251)\\ &= [210, 23, 36, 118, 122]\,.\end{aligned} \tag{139}$$

For the other ciphertexts $\overleftrightarrow{Cipher_1}\Big|^{5}$, $\overleftrightarrow{Cipher^*}\Big|^{5}$, and $\overleftrightarrow{Cipher_1^*}\Big|^{5}$ in (135) and (137), we have

$$\begin{aligned}&(\overleftrightarrow{Cipher_1}\Big|^{5} \circledast \overleftrightarrow{f}_3\Big|^{5})(\bmod 18072001)\\ &\quad = [3041577, 2642300, 3569758, 1907467, 3871797],\end{aligned} \tag{140}$$

$$\begin{aligned}&(\overleftrightarrow{Cipher^*}\Big|^{5} \circledast \overleftrightarrow{f}_3\Big|^{5})(\bmod 18072001)\\ &\quad = [4450691, 4541115, 4066487, 3590422, 3912710],\end{aligned} \tag{141}$$

$$\begin{aligned}&(\overleftrightarrow{Cipher_1^*}\Big|^{5} \circledast \overleftrightarrow{f}_3\Big|^{5})(\bmod 18072001)\\ &\quad = [3217277, 3669141, 3982904, 4102462, 3585155],\end{aligned} \tag{142}$$

getting

$$\begin{aligned}\overleftrightarrow{M_{0_1}}\Big|^{5} &= [3041577, 2642300, 3569758, 1907467,\\ &\quad 3871797](\bmod 251)\\ &= [210, 23, 36, 118, 122],\end{aligned} \tag{143}$$

$$\begin{aligned}\overleftrightarrow{M_0^*}\Big|^{5} &= [4450691, 4541115, 4066487, 3590422,\\ &\quad 3912710](\bmod 251)\\ &= [210, 23, 36, 118, 122],\end{aligned} \tag{144}$$

$$\begin{aligned}\overleftrightarrow{M_{0_1}^*}\Big|^{5} &= [3217277, 3669141, 3982904, 4102462,\\ &\quad 3585155](\bmod 251)\\ &= [210, 23, 36, 118, 122],\end{aligned} \tag{145}$$

so we have

$$\begin{aligned}\overleftrightarrow{M_0}\Big|^{5} = \overleftrightarrow{M_{0_1}}\Big|^{5} &= \overleftrightarrow{M_0^*}\Big|^{5} = \overleftrightarrow{M_{0_1}^*}\Big|^{5}\\ &= [210, 23, 36, 118, 122]\,,\end{aligned} \tag{146}$$

hence the decryption's final step is

$$\begin{aligned}\overleftrightarrow{M_1}\Big|^{5} &= \overleftrightarrow{M_0}\Big|^{5} \circledast \overleftrightarrow{F}_{251}\Big|^{5} (\bmod 251)\\ &\equiv [210, 23, 36, 118, 122]\\ &\quad \circledast [92, 223, 74, 164, 128]\,(\bmod 251)\\ &\equiv [72, 101, 108, 108, 111]\\ &= Hello.\end{aligned} \tag{147}$$

ACKNOWLEDGMENT

Shih-Lien Lu was with Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu 30075, Taiwan.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.

[2] S. Beauregard, "Circuit for Shor's algorithm using 2n+3 qubits," 2002, *arXiv:quant-ph/0205095*. [Online]. Available: https://arxiv.org/abs/quant-ph/0205095

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 212–219.

[4] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: Quantum resource estimates," in *Proc. Int. Workshop Post-Quantum Cryptogr.* Cham, Switzerland: Springer, 2016, pp. 29–43.

[5] M. Ajtai, "Generating hard instances of lattice problems," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 99–108.

[6] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. 29th Annu. ACM Symp. Theory Comput. (STOC)*, 1997, pp. 284–293.

[7] J. Buchmann and J. Ding, "Post-quantum cryptography," in *Proc. 2nd Int. Workshop PQCrypto*, 2008, pp. 17–19.

[8] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, 2009.

[9] D. J. Bernstein, "Comparing proofs of security for lattice-based encryption," *Target*, vol. 1, p. 2, Oct. 2019.

[10] J. Hoffstein, D. Lieman, J. Pipher, and J. H. Silverman, *NTRU: A Public Key Cryptosystem*. Burlington, MA, USA: NTRU Cryptosystems, 1999.

[11] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, "NTRU Prime: Reducing attack surface at low cost," in *Proc. Int. Conf. Sel. Areas Cryptogr.* Cham, Switzerland: Springer, 2017, pp. 235–260.

[12] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1006–1018.

[13] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Apr. 2018, pp. 353–367.

[14] M. Hamburg, "Post-quantum cryptography proposal: Threebears," NIST, Gaithersburg, MD, USA, Tech. Rep. 2019/0625, 2019.

[15] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. New York, NY, USA: Springer, 2009.

[16] S. Chatterjee and P. Sarkar, *Identity-Based Encryption*. New York, NY, USA: Springer, 2011.

[17] R. N. Pontaza Rodas and Y.-D. Lin, "Post-quantum asymmetric key cryptosystem with one-to-many distributed key management based on prime modulo double encapsulation," U.S. Patent 16 448 445, Apr. 2, 2020.

[18] NIST. (2021). *Post-Quantum Cryptography Standardization*. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization

[19] NIST. (2021). *Round 2 Submissions*. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-2-submissions

[20] Virtual Applications. (2021). *eBACS: ECRYPT Benchmarking of Cryptographic Systems*. [Online]. Available: https://bench.cr.yp.to/supercop.html

[21] Virtual Applications. (2021). *VAMPIRE—Virtual Applications and Implementations Research Lab*. [Online]. Available: http://hyperelliptic.org/ECRYPTII/vampire/

[22] *Liboqs*. Accessed: May 2020. [Online]. Available: https://github.com/open-quantum-safe/liboqs

[23] X. Lu, Y. Liu, D. Jia, H. Xue, J. He, Z. Zhang, Z. Liu, H. Yang, B. Li, and K. Wang, "LAC: Lattice-based Cryptosystems," *NIST PQC Round*, vol. 2, p. 4, Oct. 2019.

[24] E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, T. Pöppelmann, P. Schwabe, and D. Stebila, "NewHope: Algorithm specifications and supporting documentation," NIST, Gaithersburg, MD, USA, Tech. Rep. 1.02, Mar. 2019.

[25] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation," *NIST Post-Quantum Project*, vol. 9, p. 11, Apr. 2017.

[26] E. Alkim, "FrodoKEM learning with errors key encapsulation," NIST, Gaithersburg, MD, USA, Tech. Rep. 2017/1130, Nov. 2017.

[27] J. Howe, T. Oder, M. Krausz, and T. Güneysu. (Oct. 25, 2018). *Standard Lattice Based Key Encapsulation on Embedded Devices*. Accessed: May 2020. [Online]. Available: https://www.youtube.com/watch?v=zAfPwuBKixk

[28] (2019). *Round 2 Official Comment: Frodo*. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/official-comments/FrodoKEM-round2-official-comment.pdf

[29] J. Howe, T. Oder, M. Krausz, and T. Güneysu, "Standard lattice-based key encapsulation on embedded devices," in *Proc. IACR Trans. Cryptograph. Hardw. Embedded Syst.*, Aug. 2018, pp. 372–393.

[30] (Jul. 2019). *OFFICIAL COMMENT: Three Bears*. Accessed: May 2020. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/official-comments/Three-Bears-round2-official-comment.pdf

[31] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schanck, P. Schwabe, W. Whyte, and Z. Zhang, "NTRU algorithm specifications and supporting documentation," in *Proc. 2nd PQC Standardization Conf.*, 2019, pp. 1–5.

[32] D. J. Bernstein, T. Lange, and C. van Vredendaal. (2019). *NTRU Prime: Round 2 20190330*. [Online]. Available: http://ntruprime.cr.yp.to/nist/ntruprime-20190330.pdf

[33] (2019). *Round 2 Official Comment: Ntruencrypt & NTRU*. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-2/official-comments/NTRU-round2-official-comment.pdf

[34] North Cryptosystem. (2019). *NIST NewHope Round 2 Official Comments*. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-2/official-comments/NewHope-round2-official-comment.pdf

[35] Lattice-Based Cryptosystems. (2019). *NIST LAC Round 2 Official Comments*. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-2/official-comments/LAC-round2-official-comment.pdf

[36] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–41, Feb. 2019.

[37] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2010, pp. 1–23.

[38] D. J. Bernstein, S. Jeffery, T. Lange, and A. Meurer, "Quantum algorithms for the subset-sum problem," in *Proc. Int. Workshop Post-Quantum Cryptogr.* Berlin, Germany: Springer, 2013, pp. 16–33.

[39] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST, Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-22, 2010. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

[40] NIST. (2020). *Guide to the Statistical Tests*. [Online]. Available: https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software/Guide-to-the-Statistical-Tests.

[41] NIST. (2020). *NIST SP 800-22: Download Documentation and Software*. [Online]. Available: https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software.

[42] NIST. (2021). *Security (Evaluation Criteria)*. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria).

[43] Arduino. (2021). *Data Type: Unsigned Long*. [Online]. Available: https://www.arduino.cc/reference/en/language/variables/data-types/unsignedlong/

[44] R. Pontaza, Y.-D. Lin, S.-L. Lu, and K.-J. Chang. (Jul. 2020). *O2MD2—Official Software Repository*. Accessed: Jul. 2020. [Online]. Available: https://github.com/pontazaricardonctu/o2md2

[45] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Des., Codes Cryptogr.*, vol. 75, no. 3, pp. 565–599, Jun. 2015.

[46] J. Hoffstein, J. Pipher, J. H. Silverman, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, vol. 1. New York, NY, USA: Springer, 2008.

[47] C. Peikert, "A decade of lattice cryptography," *Found. Trends Theor. Comput. Sci.*, vol. 10, no. 4, pp. 283–424, 2016.

[48] GCC. *Double-Word Integers*. Accessed: May 2020. [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc/Long-Long.html

[49] (2019). *ROUND 2 OFFICIAL COMMENT: NTRU Prime*. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-2/official-comments/NTRU-Prime-round2-official-comment.pdf.

[50] A. Nitaj, *The Mathematics of the NTRU Public Key Cryptosystem*. Hershey, PA, USA: IGI Global, 2015.

[51] T. Kleinjung, "Factorization of a 768-bit RSA modulus (version 1.4)," *Lect. Notes Comput. Sci.*, vol. 6223, p. 20, Oct. 2010.

[52] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 1–14.

[53] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*. San Francisco, CA, USA: Morgan Kaufmann, 2013.

[54] D. V. Bailey, D. Coffin, A. Elbirt, J. H. Silverman, and A. D. Woodbury, "NTRU in constrained devices," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2001, pp. 262–272.

[55] J. H. Silverman, *The Arithmetic of Elliptic Curves*, vol. 106. New York, NY, USA: Springer, 2009.

[56] J. H. Silverman, *Advanced Topics in the Arithmetic of Elliptic Curves*, vol. 151. New York, NY, USA: Springer, 2013.

[57] J. Buchmann, *Introduction to Cryptography*. New York, NY, USA: Springer, 2013.

[58] A. Engel, *Problem-Solving Strategies*. New York, NY, USA: Springer-Verlag, 1976.

[59] R. De Prisco and M. Yung, *Security and Cryptography for Networks*. New York, NY, USA: Springer, 2006.

[60] E. Hlawka, J. Schoissengeier, and R. Taschner, *Geometric and Analytic Number Theory*. New York, NY, USA: Springer, 2012.

[61] I. Niven, H. S. Zuckerman, and H. L. Montgomery, *An Introduction to the Theory of Numbers*. Hoboken, NJ, USA: Wiley, 2013.

[62] M. Sipser, *Introduction to the Theory of Computation*, vol. 2. Boston, MA, USA: Thomson Course Technology, 2006.

[63] C. H. Papadimitriou, *Computational Complexity*. Hoboken, NJ, USA: Wiley, 2003.

[64] H. Cohen, *A Course in Computational Algebraic Number Theory*, vol. 138. Berlin, Germany: Springer, 2013.

[65] T. M. Apostol, "Some properties of completely multiplicative arithmetical functions," *Amer. Math. Monthly*, vol. 78, no. 3, pp. 266–271, Mar. 1971.

[66] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.

[67] J. G. Merchan, "Arithmetic architectures for finite fields GF (p_m) with cryptographic applications," Ph.D. dissertation, Dept. Elect. Eng. Inf. Technol., Ruhr-Univ. Bochum, Berlin, Germany, 2004. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.7051&rep=rep1&type=pdf

[68] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2004, pp. 119–132.

[69] J. Guajardo, T. Güneysu, S. S. Kumar, C. Paar, and J. Pelzl, "Efficient hardware implementation of finite fields with applications to cryptography," *Acta Appl. Math.*, vol. 93, nos. 1–3, pp. 75–118, 2006.

[70] R. N. Pontaza Rodas, Y.-D. Lin, S.-L. Lu, and K.-J. Chang. (2021). *O2MD2: A New Post-Quantum Cryptosystem With One-to-Many Distributed Key Management Based on Prime Modulo Double Encapsulation (Additional examples)*. [Online]. Available: https://drive.google.com/drive/folders/1HYXsBSPjREaSXZWUgxedy7iHVUsEHJ96

[71] G. J. Simmons, "Symmetric and asymmetric encryption," *ACM Comput. Surv.*, vol. 11, no. 4, pp. 305–330, 1979.

[72] R. M. Gray, "Toeplitz and circulant matrices: A review," *Found. Trends Commun. Inf. Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[73] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[74] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, Sep. 2002.

[75] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*. Berlin, Germany: Springer, 2010, pp. 3–37.

[76] M. Potkonjak and V. Goudar, "Public physical unclonable functions," *Proc. IEEE*, vol. 102, no. 8, pp. 1142–1156, Aug. 2014. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6856138

[77] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Proc. Int. Algorithmic Number Theory Symp.* Berlin, Germany: Springer, 1998, pp. 267–288.

[78] A. Chopra, "GLYPH: A new insantiation of the GLP digital signature scheme," *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 766, 2017. [Online]. Available: https://eprint.iacr.org/2017/766.pdf

[79] J. A. Buchmann, D. Butin, F. Göpfert, and A. Petzoldt, "Post-quantum cryptography: State of the art," in *The New Codebreakers*. Berlin, Germany: Springer, 2016, pp. 88–108.

[80] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—A new hope," in *Proc. USENIX Secur. Symp.*, 2016, pp. 327–343.

[81] D. Coppersmith and A. Shamir, "Lattice attacks on NTRU," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1997 pp. 52–61.

[82] N. Howgrave-Graham, "A hybrid lattice-reduction and meet-in-the-middle attack against NTRU," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2007, pp. 150–169.

[83] N. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos, J. H. Silverman, A. Singer, and W. Whyte, "The impact of decryption failures on the security of NTRU encryption," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2003, pp. 226–246.

RICARDO NEFTALI PONTAZA RODAS received the master's degree in applied mathematics from National Taiwan University, where he is currently pursuing the Ph.D. degree in computer science. He is also pursuing the Ph.D. degree in electrical engineering with National Chiao Tung University. He is also a former international mathematical competitor and a former stock exchange broker, have experience with algebra, number theory, and cryptography.

YING-DAR LIN (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), in 1993. He was a Visiting Scholar at Cisco Systems, San Jose, from 2007 to 2008, the CEO at Telecom Technology Center, Taiwan, from 2010 to 2011, and the Vice President of the National Applied Research Labs (NARLabs), Taiwan, from 2017 to 2018. He co-founded L7 Networks Inc., in 2002, later acquired by D-Link Corporation. He also founded and directed Network Benchmarking Lab (NBL), in 2002, which reviewed network products with real traffic and automated tools, also an approved test lab of the Open Networking Foundation (ONF), and spun off O'Prueba Technology Inc., in 2018. He is currently a Chair Professor of computer science with National Chiao Tung University (NCTU), Taiwan. His research interests include machine learning for cybersecurity, wireless communications, network softwarization, and mobile edge computing. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He published a textbook *Computer Networks: An Open Source Approach* (McGraw-Hill, 2011), with Ren-Hung Hwang and Fred Baker. He is a Distinguished Lecturer of IEEE (2014–2017), an ONF Research Associate (2014–2018), and received the K. T. Li Breakthrough Award, in 2017, and the Research Excellence Award, in 2017 and 2020. He has served or is serving on the editorial boards for several IEEE journals and magazines, including the Editor-in-Chief of IEEE COMMUNICATIONS SURVEYS & TUTORIALS (COMST) with impact factor increased from 9.22 to 23.7 during his term in 2017–2020.

SHIH-LIEN (LINUS) LU received the B.S. degree in electrical engineering and computer science from UC Berkeley and the M.S. and Ph.D. degrees in computer science and engineering from UCLA.

He is currently the Chief Solutions Officer at PieceMakers Technology. He was the Director of TSMC Research and Development, from 2016 to 2021. From 1999 to 2016, he was with Intel Corporation, Hillsboro, OR, USA, where he was a Research Scientist and the Research Group Manager. And later, he was also the Director of the Memory Architecture Lab, Intel Labs. He served on the faculty for the ECE Department, Oregon State University, as an Assistant Professor, from 1991 to 1995, and as a tenured Associate Professor, until 2001 (on leave the last two years). From 1984 to 1991, he worked on the MOSIS project at USC/ISI, which provides U.S. research and education community VLSI fabrication services. He has published more than 100 articles and authored or coauthored more than 150 U.S. patents. His research interests include computer architecture, memory system and circuits, low power VLSI design, and hardware security.

KEH-JENG CHANG received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree in computer science from the University of California at Los Angeles, USA. After receiving his degree from UCLA, he spent more than 14 years conducting VLSI electronic design automation (EDA) researches in the Silicon Valley in Northern California for two companies consecutively, i.e., Hewlett-Packard Company and Sequence Design Inc. Then, he returned to his home country and continued his VLSI EDA researches at National Tsing Hua University and Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan. He has published scores of journal articles and conference papers and has been awarded more than 15 patents, in USA and Asia, on methods and systems targeting yield improvement for 3-D nanometer devices and 3-D electronic packaging.