

Received 12 October 2025, accepted 22 November 2025, date of publication 2 December 2025,
date of current version 11 December 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3639247

RESEARCH ARTICLE

Accelerating ALTEQ With Hardware–Software Co-Design on FPGA

YONGRYEOL CHOI¹, (Student Member, IEEE), JAESEOK LEE¹, (Student Member, IEEE),
YOUNGBEOM KIM², (Student Member, IEEE), WILLY SUSILO³, (Fellow, IEEE),
OKYEON YI¹, (Member, IEEE), ILSUN YOU¹, (Senior Member, IEEE),
AND SEOG CHUNG SEO¹, (Member, IEEE)

¹Department of Cyber Security, Kookmin University, Seoul 02707, South Korea

²Department of Financial Information Security, Kookmin University, Seoul 02707, South Korea

³Institute of Cybersecurity and Cryptology (iC2), University of Wollongong, Wollongong, NSW 2500, Australia

Corresponding author: Seog Chung Seo (scseo@kookmin.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant by Korean Government through the Ministry of Science and ICT (MSIT) (A Study on Post-Quantum Cryptography (PQC) Optimization and Security Protocol Migration to Neutralize Advanced Quantum Attacks in Beyond 5G-Based Next-Generation IoT Computing Environments) under Grant 2022-00207416 (25%), in part by the National Research Foundation of Korea (NRF) Grant funded by Korean Government (MSIT) under Grant 2022R1C1C1013368 (25%), and in part by the Basic Science Research Program through NRF funded by the Ministry of Education under Grant RS-2025-25411243 (50%).

ABSTRACT The emergence of quantum computing poses a significant threat to the security of modern public-key cryptography, spurring a global effort to standardize Post-Quantum Cryptography (PQC). However, optimization research has predominantly focused on software implementations and lattice-based algorithms, leaving a notable gap in the hardware acceleration of non-lattice-based schemes, particularly on platforms such as FPGAs. This paper aims to bridge this gap by proposing the first HW/SW co-design to accelerate ALTEQ, a non-lattice-based digital signature algorithm submitted to the NIST PQC additional signature competition. Our approach maximizes performance by strategically offloading ALTEQ's most computationally intensive operations—the hash function and modular arithmetic—to dedicated hardware accelerators. To the best of our knowledge, this is the first study to investigate a hardware acceleration architecture for ALTEQ or similar schemes. The proposed design achieves performance improvements of up to approximately 88% for Keygen, 91% for Sign, and 92% for Verify. This demonstrates that algorithms not selected for standardization can still achieve substantial performance gains through hardware optimization, suggesting the potential for re-evaluating such schemes for specific application environments and encouraging further research into hardware implementations of a broader range of PQC algorithms.

INDEX TERMS Post-quantum cryptography, ALTEQ, HW/SW co-design, FPGA.

I. INTRODUCTION

With the proposal of Shor's algorithm [1] in 1994, it became known that modern public-key cryptosystems based on the discrete logarithm and integer factorization problems could be solved in polynomial time in a quantum computing environment. Furthermore, with recent advancements in quantum computing technology, quantum computers are no longer merely theoretical but are expected to become a reality in the near future. If this technology is commercialized, current cryptographic systems will no longer be

secure. Consequently, active research has been conducted on Post-Quantum Cryptography (PQC), which comprises cryptosystems designed to operate securely even in a quantum computing environment. Since 2017, National Institute of Standards and Technology (NIST) has held a PQC competition [2] to select standardization algorithms, and in 2024, two Key Encapsulation Mechanisms (KEMs) and two Digital Signature Algorithms (DSAs) were selected as the final algorithms. Additionally, a fourth round for KEMs and an additional round for DSAs were conducted to select more schemes, and a final KEM algorithm has now been chosen. South Korea has also been running its KpqC competition [3] since 2021 to select national standard PQC algorithms, with

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamad Afendee Mohamed¹.

two KEMs and two DSAs selected in 2025, similar to the NIST process.

Extensive research has been conducted to optimize the algorithms submitted to these competitions for faster performance while achieving quantum security across all devices [4], [5], [6]. In fact, significant research has also been conducted on algorithms not selected for final standardization, with implementations on embedded devices such as Cortex-M4 [7], [8] and RISC-V [9], or in SIMD environments like AVX2/512 [10], [11]. However, compared to these environments, relatively little research has been conducted on optimized implementations for hardware devices such as FPGAs. Moreover, compared to lattice-based algorithms, those based on other foundations have been the subject of significantly fewer studies.

To address this research gap and explore the hardware acceleration potential of non-lattice-based algorithms, we investigated candidates from the ongoing NIST Additional DSA round, focusing on schemes that have been less studied. Among them, ALTEQ consumes most of its computational cycles in hash function processing and modular arithmetic, indicating a high potential for performance improvement if implemented with a HW/SW co-design. We are the first to propose a HW/SW co-design for performance acceleration targeting ALTEQ [12], which was submitted in Round 1. To the best of our knowledge, no prior research has been conducted on performance acceleration for ALTEQ or similar schemes. Our proposed architecture can accelerate the entire ALTEQ scheme up to approximately 88% with low-area usage. This demonstrates that even an algorithm that did not advance to Round 2 possesses significant potential for performance acceleration. Through this, we can create an opportunity for cryptographic schemes not selected for standardization to be re-evaluated for their potential in specific application environments.

A. CONTRIBUTION

The contributions of this paper are as follows: 1. We are the first to propose a HW/SW co-design implementation methodology for ALTEQ, thereby accelerating its performance. While the proposed module is specialized for ALTEQ, it is designed to be easily reconfigurable for the modular arithmetic of other algorithms. 2. We present profiling results for the performance acceleration of ALTEQ and identify its performance bottlenecks. 3. We provide performance metrics of our implementation to demonstrate the performance improvement of ALTEQ, and we analyze its differences from and limitations compared to existing cryptosystems. We have confirmed that the proposed design improves ALTEQ's performance up to approximately 88%, 91%, and 92% for Keygen, Sign, and Verify respectively.

II. RELATED WORK

The mainstream of PQC-related hardware research conducted to date can be broadly divided into two categories: designing external hardware accelerators to offload specific operations, and extending a processor's Instruction Set Architecture

(ISA) to support acceleration. The first approach, designing external FPGA-based accelerators, focuses on implementing the computational bottlenecks of PQC algorithms as dedicated hardware modules to enhance overall performance. For instance, Mao et al. [13] accelerated the core operations of CRYSTALS-Dilithium, such as the Number Theoretic Transform (NTT) and Keccak, thereby reducing signing and verification times. Similar studies can be found for various other algorithms. Xing et al. [14] optimized the performance of CRYSTALS-Kyber by accelerating its main operations and designing a hardware implementation of the Modified Barrett Reduction technique. Pendyala et al. [15] significantly improved sampling performance by implementing Falcon's core Gaussian Sampler process in hardware. Likewise, Dolmeta et al. [16] designed accelerators for operations like Karatsuba multiplication and Keccak to improve the performance of HQC, a code-based PQC scheme. These studies demonstrate the effective resolution of performance bottlenecks in specific algorithms.

The second approach has been predominantly researched in processor environments where the ISA is easily extensible, such as RISC-V. This method involves adding new instructions required for PQC operations, allowing software to directly control hardware acceleration units. Fritzmann et al. [17] designed operation modules common to PQC, such as NTT, modular arithmetic, and Keccak, and proposed custom RISC-V instructions to control them, reporting performance improvements for several algorithms, including Kyber and NewHope. Alkim et al. [18] improved the performance of Kyber and NewHope through their research on RISC-V ISA extensions for finite field arithmetic and Barrett Reduction. This ISA extension approach offers advantages over external accelerators, such as lower data transfer overhead and tighter integration with software.

However, all these studies share a common focus: they target either finalists of the NIST PQC Competition or lattice-based cryptosystems, which are currently the subject of extensive optimization efforts. Acceleration research for algorithms based on other mathematical problems remains limited. Therefore, research that identifies the potential for significant performance improvement in less-studied algorithms, and then validates this through actual implementation, can suggest paths to practical application for unexplored PQC schemes and contribute to broadening the scope of the PQC hardware research field. Consequently, we target ALTEQ, for which no acceleration research has yet been conducted, to propose an efficient hardware accelerator architecture for its core operations and evaluate its performance.

III. PRELIMINARY

In this section, we provide preliminaries to understand our methodologies, which will be given in Sections III and IV.

A. NOTATIONS

We describe the notation used throughout this paper. $s \leftarrow_R S$ denotes that s is randomly sampled in the set S . n denotes the vector space dimension, and q denotes the finite field order.

TABLE 1. The detailed parameters of ALTEQ.

Mode	Parameters	Parameter Set (n, q, r, K, C)	Private Key Size	Public Key Size	Signature Size
Balanced	I	(13, $2^{32} - 5$, 84, 22, 7)	32	8040	15928
	III	(20, $2^{32} - 5$, 201, 28, 7)	48	31968	49048
ShortSig	I	(13, $2^{32} - 5$, 16, 14, 458)	32	523984	9560
	III	(20, $2^{32} - 5$, 39, 20, 229)	48	1044288	32552

$\mathbb{F}_q$ denotes the finite field of order q , and $\mathbb{F}_q^n$ denotes the vector space of length- n column vectors over $\mathbb{F}_q$. $C + 1$ denotes the number of alternating trilinear forms. For matrix A , A^T denotes its transpose. For matrix $A \in \mathbb{F}^{n \times n}$, A is a column matrix if and only if A is the form

$$A = \begin{pmatrix} 1 & \dots & 0 & c_1 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & c_{i-1} & 0 & \dots & 0 \\ 0 & \dots & 0 & c_i & 0 & \dots & 0 \\ 0 & \dots & 0 & c_{i+1} & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & c_n & 0 & \dots & 1 \end{pmatrix}$$

for $i \in [1, n]$. An alternating trilinear form (ATF) on $\mathbb{F}_q^n$ is a map $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ such that $\phi(u, u, v) = \phi(u, v, u) = \phi(v, u, u) = 0$ for $u, v \in \mathbb{F}_q^n$. For general linear group $GL(n, q)$ of degree n over $\mathbb{F}_q$, a group action $\circ$ on ATF ϕ is defined as follows: $(A, \phi) \mapsto \phi \circ A : (u, v, w) \mapsto \phi(A^T u, A^T v, A^T w)$ for all $u, v, w \in \mathbb{F}_q^n$. $\circ$ derives an equivalence relation $\sim$ on ATF ϕ if and only if there exists $A \in GL(n, q)$ such that $\phi = \psi \circ A$. This relation is denoted as $\phi \sim \psi$. Lastly, H denotes the cryptographic hash function; ALTEQ uses SHA-3 family for its hash function.

B. ALTERNATING TRILINEAR FORM EQUIVALENCE

The Alternating Trilinear Form Equivalence (ATFE) problem is an algorithmic problem about ATFs. There are some extended versions of ATFE, so we introduce them, which is necessary for ALTEQ.

- **The ATFE Problem:** For given two ATFs $\phi, \psi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$, find $A \in GL(n, q)$ such that $\phi = \psi \circ A$ if it exists.
- **The psATFE Problem:** For given two ATFs $\phi, \psi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ such that $\phi \sim \psi$, find $A \in GL(n, q)$ such that $\phi = \psi \circ A$. This is the promise search version of ATFE since it guarantees $\phi \sim \psi$.
- **The C-psATFE Problem:** For given C ATFs $\phi_1, \dots, \phi_C : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ such that $\phi_i \sim \phi_j$ for any $i, j \in [1, C]$, find $A \in GL(n, q)$ such that $\phi_i = \phi_j \circ A$.

In quantum computing, ATFE and psATFE have a computational complexity of $\mathcal{O}(q^{n/3})$, by Szegedy's algorithm [19] and Tani's algorithm [20] respectively.

C. ALTEQ SPECIFICATION

ALTEQ, a candidate submitted to Round 1 of the NIST Additional DSA competition, is a digital signature algorithm

Algorithm 1 KeyGen Function of ALTEQ

Inputs: The variable number $n \in \mathbb{N}$, a prime q , the ATF number $C + 1$

Output: Public key pk and private key sk

- 1: $pk \leftarrow C$ matrices $\{A_i\}_{i=0}^{C-1}$ such that $\phi_i \circ A_i = \phi_C$
- 2: **for** $i \in \{0, \dots, C - 1\}$ **do**
- 3: $\phi_i \leftarrow \phi_C \circ A_i$
- 4: **end for**
- 5: **for** $i \in \{0, \dots, C - 1\}$ **do**
- 6: $A_i \leftarrow A_i^{-1}$
- 7: **end for**
- 8: **return** $pk = \{\phi_i\}_{i=0}^C, sk = \{A_i\}_{i=0}^{C-1}$

Algorithm 2 Sign Function of ALTEQ

Inputs: Public key $\{\phi_i\}_{i=0}^C \in \text{ATF}(n, q)$. Private key $\{A_i\}_{i=0}^{C-1} \in GL(n, q)$. Message M . $r, C, \lambda \in \mathbb{N}$.

Output: Signature S of M

- 1: **for** $i \in \{0, \dots, r - 1\}$ **do**
- 2: $B_i \leftarrow_R GL(n, q)$.
- 3: $\psi_i \leftarrow \phi_C \circ B_i$.
- 4: **end for**
- 5: $\text{cha} \leftarrow H(M|\psi_0|\dots|\psi_{r-1}) \in \{0, 1\}^{2\lambda}$
- 6: $\{b_i\}_{i=0}^{r-1} \leftarrow \text{Expand}(\text{cha})$
- 7: **for** $i \in \{0, \dots, 1\}$ **do**
- 8: $D_i \leftarrow A_{b_i} B_i$
- 9: **end for**
- 10: **return** $S = (\text{cha}, D_0, \dots, D_{r-1})$

based on ATFE. It adopts a signature structure that combines the Goldreich-Micali-Wigderson (GWM) protocol [21] with the Fiat-Shamir transformation. To achieve EUF-CMA security, it additionally uses C-psATFE, one of the various ATFE variants. ALTEQ provides two parameter sets, I and III, each of which offers a Balanced and a ShortSig mode. The Balanced mode features small public key and signature sizes, whereas the ShortSig mode provides a shorter signature size at the cost of a significantly larger public key. For both modes, the private key size remains the same for a given parameter set. The sizes of the public key, private key, and signature for each parameter set and mode of ALTEQ are presented in Table 1.

D. HW/SW CO-DESIGN

In the fields of embedded systems and high-performance computing, heterogeneous computing systems, which utilize

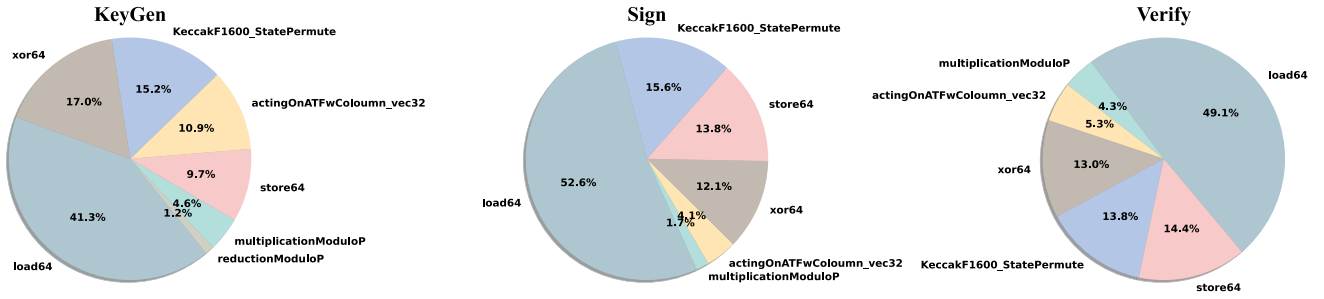


FIGURE 1. The profiling results of ALTEQ Top-level functions: KeyGen, Sign, Verify.

Algorithm 3 Verify Function of ALTEQ

Inputs: The Public Key $pk = \{\phi_i\}_{i=0}^C$, the signature $S = (\text{cha}, D_0, \dots, D_{r-1})$, $b_i \in \{0, \dots, C\}$, $D_i \in GL(n, q)$, the message M

Output: *Success* if S is a valid signature of M , *Failed* otherwise

```

1: for  $i \in \{0, \dots, r-1\}$  do
2:    $\psi'_i \leftarrow \phi_{b_i} \circ D_i$ 
3: end for
4:  $\text{cha}' = H(M | \psi'_0 | \dots | \psi'_{r-1}) \in \{0, 1\}^{2\lambda}$ 
5: if for every  $i \in \{0, \dots, r-1\}$ ,  $b_i = b'_i$  then
6:   return Success
7: else
8:   return Failed
9: end if

```

multiple types of processing units, are now widely used to overcome the limitations of homogeneous systems composed of a single processor type. Prominent examples include systems that use a CPU and GPU in tandem, or System-on-Chip (SoC) FPGAs, which are the main focus of this paper. In particular, modern SoC FPGAs such as Xilinx's Zynq UltraScale+ MPSoC feature an integrated architecture with a Processing System (PS) for software processing and Programmable Logic (PL) for hardware acceleration on a single chip. The descriptions of the PS and PL are as follows:

- **PS:** This is the area that includes microprocessor-based cores such as ARM or RISC-V, as well as memory controllers and standard peripheral interfaces. It is primarily used to control the overall system and execute software code that handles complex algorithms or sequential logic.
- **PL:** This area possesses the characteristics of a classical FPGA, composed of numerous configurable logic blocks (CLBs), DSP slices, and Block RAMs (BRAMs). Developers can reconfigure these resources to design and program accelerators for specific operations. Such accelerators can be designed for high parallelism or high throughput, achieving significant performance improvements compared to software execution.

This PS-PL architecture is fundamental to HW/SW co-design, as it leverages the advantages of both software

flexibility and hardware performance. However, the transmission and reception of data and signals between these two domains are essential for their operation. The Advanced eXtensible Interface (AXI) protocol facilitates seamless communication between the PS and PL by enabling efficient and reliable data transfer. The primary types of the AXI protocol are as follows:

- **AXI4 (Full AXI):** A memory-mapped protocol aimed at high-bandwidth, low-latency communication. It supports high-efficiency communication through features like burst transfers, which can move up to 256 consecutive data beats in a single transaction, and out-of-order completion, which allows receiving data from slave modules that can respond quickly, regardless of the request order.
- **AXI-Lite:** A simplified version of AXI4 used for accessing simple control and status registers where complex burst transfers are unnecessary. It does not support bursts and has a fixed data width of 32 or 64 bits.
- **AXI-Stream:** A protocol for unidirectional, continuous streaming without the concept of addresses, unlike memory-mapped protocols. It is efficient for continuously sending large amounts of data from one point to another and is suitable for applications like large-scale data transfers via DMA and real-time data stream processing.

In a co-design environment, a developer maps the hardware accelerator designed in the PL to the PS's memory space via an AXI interface. This allows software running on the PS to control the hardware accelerator, exchange data, and utilize it as simply as accessing a specific memory address.

IV. PROFILING ALTEQ

In this section, we identify the performance bottlenecks of the ALTEQ reference code through profiling results. When accelerating an algorithm, it is a common practice to identify and improve the most significant performance bottlenecks through profiling, a methodology we also adopt. For profiling ALTEQ, we used a PC equipped with an Intel i7-13700K CPU and compiled the code with the -O3 optimization flag. The profiling results, measured using the gprof tool after

executing the KeyGen, Sign, and Verify functions 1,000 times each, are presented in Figure [].

As seen in all parts of Figure [], the functions `load64`, `store64`, `xor64`, and `KeccakF1600_StatePermutate` are all related to the SHA-3 family. Their computational load accounts for over 80% of the total execution time, indicating a very high overhead. Therefore, these functions should be the primary candidates for acceleration. The next most computationally intensive functions are `multiplicationModuloP` and `reductionModuloP`, which together account for approximately 5% of the total operations. Although this is considerably less than the load from the SHA-3 family, we selected them as acceleration targets considering that their relative computational load will increase once the SHA-3 family functions are optimized. These functions are called repeatedly within double or triple loops in the `actingOnATFwColoumn_vec32` function.

The group action, denoted by in the `Sign` function of ALTEQ as shown in Algorithm 2, corresponds to the `actingOnATFS` function in the implementation. This function, in turn, repeatedly calls `actingOnATFwColoumn_vec32`, further increasing the computational load. While the cost of a single modular operation is not high, its repeated invocation means that reducing its execution time can contribute to overall performance acceleration. These group action operations can also be found in Algorithm 1, lines 2-4, and Algorithm 3, lines 1-3. The `actingOnATFwColoumn_vec32` function is required for all operations denoted by the group action, confirming that modular arithmetic is used extensively throughout ALTEQ's KeyGen, Sign, and Verify procedures.

V. HARDWARE DESIGN

In this section, we propose a unified hardware unit for modular multiplication and subtraction, designed to accelerate the performance of ALTEQ. Furthermore, we describe the architecture of a previously proposed SHA-3 family accelerator and present a new architecture that integrates our unified modular unit into this structure.

A. OVERALL DESIGN

The overall design is targeted for an ARM microprocessor-based Xilinx Zynq SoC. As the proposed design is a HW/SW co-design, it utilizes both the Processing System (PS) and Programmable Logic (PL) domains. The corresponding data flow is as follows:

- **PS-PL Data Transfer:** The ARM processor in the PS initializes the Direct Memory Access (DMA) according to predefined settings and sends the data stored in DDR memory. This data is processed by the hardware modules and is transferred via the AXI-Stream protocol. In this step, control signals for module selection and operation are also transmitted via AXI-Lite.

- **Data Buffering:** The transmitted data moves from the DMA to an internal IP, the AXI Input FIFO, where it is buffered until the hardware module begins operation.
- **Operating Hardware Module:** The buffered data is sent to the target module for execution. The Keccak unit has an additional buffering stage due to its dedicated inner FIFO, whereas the modular unit does not have one.
- **PL-PS Data Transfer:** The data processed by each module is stored sequentially in the AXI Write FIFO, awaiting transfer back to the PS domain. After the transfer to the AXI Write FIFO is finished, the data is written back to the DDR memory in the PS via the DMA.

This structure enables seamless data transfer between the hardware and software. In particular, the dual-FIFO structure of the Keccak unit, which will be described later, allows for more efficient data processing. The overall design is depicted in Figure 2. The black lines represent the path for the data to be processed, while the red lines indicate the path for the control signals that manage the entire module. The Register Sets component consists of four registers that store pre-configured values transmitted from the software to operate the hardware modules. The Control Module is a Finite State Machine (FSM) that manages the entire design, and both the Keccak and Modular modules internally contain their own separate FSMs. It should also be noted that the FIFOs within the Keccak module are dedicated FIFOs, not AXI Data FIFOs.

B. FINITE STATE MACHINE

This section provides a detailed description of the Finite State Machines (FSMs) designed to precisely control the data flow of the Keccak and modular arithmetic accelerator modules. In particular, we adopted FSM structures optimized for the characteristics of each module to resolve potential data bottlenecks in the pipeline architecture and to maximize throughput.

The FSM for the Keccak module is designed with separate input and output control FSMs to manage data input and output independently. This dual FSM structure maximizes the efficiency of the data pipeline and ensures stable operation. The Keccak input control FSM consists of four states: `IDLE`, `Read_state`, `Hold_state`, and `DONE`. In the `Read_state`, data is transferred from the Keccak Read FIFO to the Keccak core, and this transfer continues for a specified input length (`din_len`). The `Hold_state` is triggered when the Keccak core is busy and cannot process new data, activating a hold signal to temporarily pause data input. This effectively prevents data loss and manages the pipeline. When the Keccak core becomes ready, the hold signal is de-asserted, and the FSM returns to the `Read_state` to resume data processing.

The Keccak output control FSM is composed of three states: `IDLE`, `Read_state`, and `DONE`. It transitions to the `Read_state` when the `rd_start` signal is triggered. In the `Read_state`, data is sequentially read from the

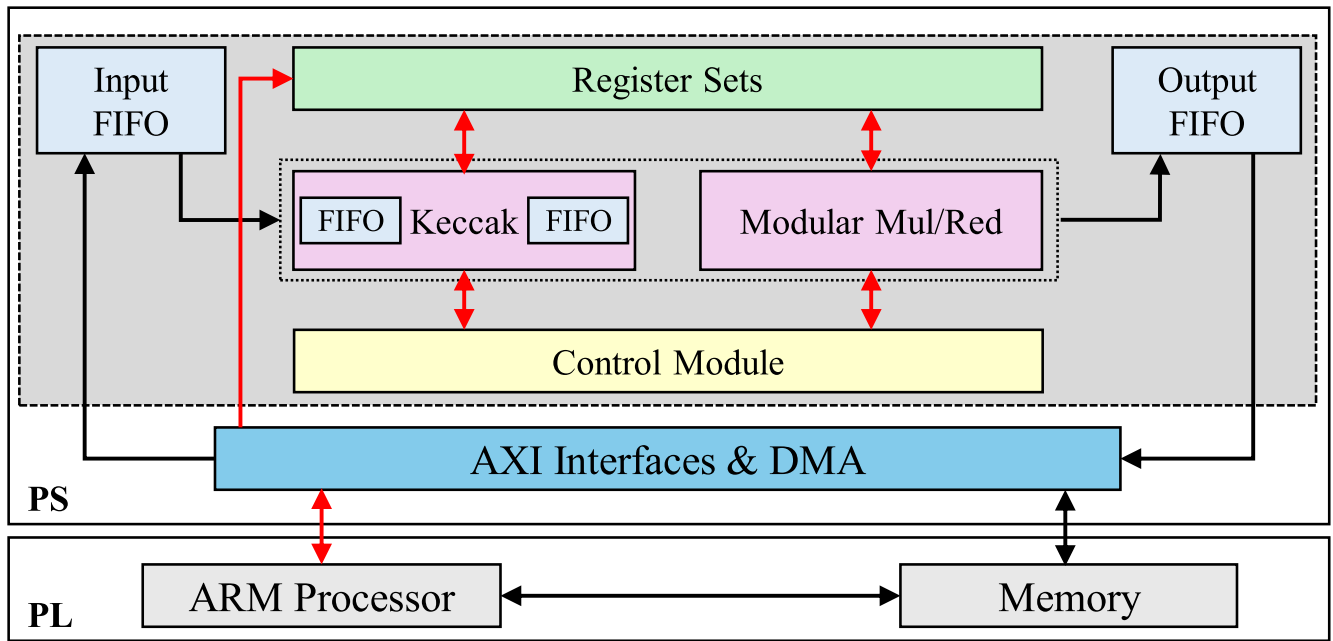


FIGURE 2. Top-level view of the proposed hardware structure.

Keccak Output FIFO for a specified output length (`rd_len`). Upon completion of the read operation, the FSM transitions to the `DONE` state. This Keccak FSM design is based on the implementation from [13].

For the modular multiplication operation, both the input operands and the output result communicate directly with the AXI Data FIFO via the AXI-Stream interface. Because the data flow is simple and unidirectional, not requiring complex flow control, a single FSM structure is sufficient, unlike the dual FSM structure of the Keccak module. Therefore, the FSM for the modular multiplier is concisely designed with three states: `IDLE`, `RUN`, and `DONE`. When the `modular_start` signal is triggered, it transitions to the `RUN` state to begin the modular operation. The last data element for the modular unit is identified by the last signal from the AXI Data FIFO. Once the final data is processed and transferred to the AXI Data FIFO, the FSM transitions to the `DONE` state.

C. ACCELERATING KECCAK

Keccak is the most significant computational bottleneck in ALTEQ, accounting for 80-90% of the total execution time, as described in Section IV. Accelerating Keccak can drastically improve the performance of ALTEQ. To achieve this, we enhance the module previously proposed by Mao et al. [13]. This module is divided into several sub-modules:

- **Keccak Read/Write FIFO:** These are dedicated FIFOs created for the Keccak module. They handle the input and output of the module and are connected to the AXI Data FIFO for data transmission and reception.

- **Read Control Unit:** This unit reads data from the Keccak Read FIFO in 64-bit chunks. It also halts the read operation if the module is busy to prevent incorrect data processing, with the overall flow being controlled by an internal FSM.
- **Keccak Core Unit:** This is the module that computes the Keccak f-1600 permutation. It consists of an internal Input Padder that performs data padding and a Permutation module that executes the f-1600 permutation. Padding is completed in 17 or 21 clock cycles depending on the data length, and the entire permutation takes 48 clock cycles. The implementation of the f-1600 permutation used here is from OpenCores.

The design by Mao et al. is specialized for the CRYSTALS-Dilithium algorithm, which presents direct compatibility issues with ALTEQ. CRYSTALS-Dilithium is a lattice-based digital signature algorithm, and its hash function usage patterns and data sizes are fundamentally different from those of ALTEQ. The most critical limitation is that the `byte_write_SHA` variable in the top control unit is restricted to 10 bits. This limits the processable data size to a maximum of 1024 bytes, which is sufficient for the typical operational patterns of CRYSTALS-Dilithium but markedly insufficient for processing the up to 13 KB of data required by ALTEQ. This size restriction causes a severe overflow during ALTEQ's large-scale hash operations, resulting in the generation of incorrect hash values and the malfunction of the entire algorithm.

Therefore, a key contribution of this research was to redesign and extend the existing architecture to systematically resolve this compatibility issue. Specifically,

D. ACCELERATING MODULAR MULTIPLICATION & REDUCTION

Algorithm 4 Modular Multiplication in ALTEQ

Output: ab reduced in 33-bits

- | | |
|--|-------------------------------------|
| 1: $r \leftarrow ab$ | $\triangleright r$ is 64-bit length |
| 2: $r \leftarrow r - q * (r \gg \log q)$ | $\triangleright \log q = 32$ |
| 3: return r | |

Inputs: 64-bit constant a . Prime number q

```

1:  $r \leftarrow a$   $\triangleright r$  is 64-bit length
2:  $r \leftarrow a - q * (a \gg \log q)$   $\triangleright \log q = 32$ 
3:  $r \leftarrow r - q * (r \gg \log q)$ 
4: return  $r_{low}$ 

```

Output: a reduced in 32-bits

- ```

1: MVN IP, #4 ▷ load prime
2: UMULL R2, R3, R1, IP ▷ $q(r \gg \log q)$
3: SUBS R0, R0, R2 ▷ $r - = q(r \gg \log q)$
4: SBC R1, R1, R3
5: UMULL R2, R3, R1, IP ▷ $q(r \gg \log q)$
6: SUBS R0, R0, R2 ▷ $r - = q(r \gg \log q)$
7: BX LR

```

The diagram illustrates the proposed algorithm for computing the product of two integers  $a$  and  $b$  using a single DSP. The inputs are  $a$ ,  $b$ , and  $q$ . The process involves several stages of bit-shifting and addition:

- Input  $a$  is shifted right by  $\log q$  bits (indicated by the trapezoid labeled  $a \gg \log q$ ) and then added to  $b$  (indicated by the circle with a plus sign).
- The result of the first addition is then added to  $a$  (indicated by the circle with a plus sign).
- The result of the second addition is then added to  $b$  (indicated by the circle with a plus sign).
- The final result is then shifted right by  $\log q$  bits (indicated by the trapezoid labeled  $\gg \log q$ ) to produce the final output.

The core idea of this unified design is to leverage the mathematical structure of the two algorithms. In Algorithm 4, line 2 computes  $r - q \lfloor r/q \rfloor$ , while in Algorithm 5, lines 2 and 3 calculate  $a - q \lfloor a/q \rfloor$  and  $r - q \lfloor r/q \rfloor$ , respectively. Both algorithms follow the same pattern of approximating the quotient, multiplying it by the modulus, and subtracting the result from the original value. Leveraging this structural similarity, we propose a unified modular unit capable of performing both operations within a single pipeline. As a result, we optimized the design to use only two DSPs, whereas a separate implementation for each operation would have required a total of three. This is a significant improvement, considering the limited DSP resources on an

**TABLE 2.** Performance results of ALTEQ-fe1 (in kilo clock cycles).

|                                    | Keygen  | Sign   | Verify | FF     | LUT    | DSP | BRAM | Fmax(MHz) |
|------------------------------------|---------|--------|--------|--------|--------|-----|------|-----------|
| Baseline <sup>(1)</sup>            | 674,421 | 87,442 | 33,915 | -      | -      | -   | -    | -         |
| Keccak HW <sup>(2)</sup>           | 152,985 | 7,786  | 4,738  | 11,863 | 10,253 | 6   | 46   | 100       |
| Keccak + Modular HW <sup>(3)</sup> | 80,494  | 5,772  | 2,723  | 12,107 | 10,350 | 8   | 46   | 100       |
| Mao et al. <sup>(4)</sup>          | 1.10ms  | 5.93ms | 1.17ms | 13,128 | 11,556 | 4   | 14   | 161       |

FPGA. Similarly, the number of subtractors was reduced from three to two, which also saved LUT resources. The operation mode is determined by a control signal that governs internal MUXes to selectively route the input values  $a$ ,  $b$ , and  $p$  according to the specific operation. For example, in modular multiplication, the inputs  $a$  and  $b$  are used directly without preprocessing. In contrast, for modular subtraction, the value  $a \gg \log q$  is selected via the control signal to perform the operation in line 2 of Algorithm 5. Since bit-shift operations can be efficiently implemented in hardware as simple wire indexing with no latency, this additional logic does not affect the overall latency. Furthermore, considering that the modular subtraction in line 2 requires both the original and the shifted value of  $a$  simultaneously, we placed an additional register to preserve the value of  $a$ , thereby resolving data dependencies and ensuring correct computational results.

As this unit is a dedicated accelerator for the ALTEQ algorithm, we optimized the bit-width of all input and output parameters to match the algorithm's requirements. The input  $a$ , used in both multiplication and subtraction, was set to a maximum of 63 bits, while a 32-bit width was sufficient for input  $b$ , which is used only in multiplication. The prime modulus  $q$  in ALTEQ is 32 bits, and this value is stored in a pipeline register and maintained every clock cycle. The output port was designed with a width of 33 bits, which is wide enough to represent the results of both multiplication (32 bits) and subtraction (33 bits) without any loss of data. This precise bit-width optimization prevents unnecessary waste of hardware resources while ensuring that all operational cases are handled safely.

A key consideration in our pipeline design is the optimization of the critical path. In the first stage, the operands are selected according to a control signal. The second stage performs a multiplication operation, which utilizes a DSP. The third stage involves a subtraction and a second multiplication, and the final stage outputs the result of the operation. Each stage was designed to have a balanced delay.

The final implementation of the unified modular unit uses a total of 8 registers, 2 DSPs, and 2 subtractors, and it has a total latency of 4 clock cycles. In contrast, when implemented in assembly on our target Zynq SoC, which does not support SIMD instructions, modular multiplication and subtraction require 6 and 7 clock cycles, respectively. An example of the assembly implementation for subtraction is provided in Algorithm 6. The proposed modular unit accelerates the performance of ALTEQ, which requires a vast number of modular operations, by reducing the latency of

these operations by up to 3 cycles. The core circuit diagram of the proposed unit is depicted in Figure 3. In the diagram, the diamond shapes represent DSPs, the blue boxes denote registers, and the symbol signifies a subtractor. Thanks to this fully-pipeline architecture, the unit achieves a high throughput, outputting one result per clock cycle after an initial 4-cycle latency. Furthermore, the unified design eliminates any additional overhead when switching between the two operations, providing consistent performance acceleration even with ALTEQ's specific operational patterns.

## VI. RESULT

### A. BENCHMARKING SETUP

To measure the performance of ALTEQ, we use a Xilinx Zedboard equipped with an XC7Z020-CLG484 FPGA and an ARM Cortex-A9 microprocessor. We used Vivado 2020.02 for IP and block design synthesis, and Vitis 2020.02 for software integration and execution. The performance was measured using the -O3 optimization flag, and we present the results for Parameter Set I of the *ShortSig* mode. The measurement results are presented in Table 2, where the Dilithium implementation by Mao et al. [13] is included for comparison.

### B. PERFORMANCE RESULTS

Baseline<sup>(1)</sup> represents the performance on the FPGA without any accelerator support. Below this, we provide the performance metrics when using only the Keccak accelerator<sup>(2)</sup>, and when using both the Keccak and modular accelerators<sup>(3)</sup>. It is important to note that the performance for the combined accelerator case<sup>(3)</sup> is a theoretical estimate, not an actual measurement from execution. This figure is an estimate obtained by taking the Keccak-only performance, subtracting the software clock cycles for modular operations, and adding the hardware clock cycles for the modular unit, including data transfer overhead. As anticipated in Section IV, accelerating Keccak alone dramatically improves the performance of ALTEQ. The performance of KeyGen, Sign, and Verify improved by approximately 77%, 91%, and 86%, respectively. Furthermore, employing the modular arithmetic unit can accelerate the performance even more, showing improvements of approximately 47%, 26%, and 42% for KeyGen, Sign, and Verify respectively, compared to the Keccak-only acceleration<sup>(2)</sup>. This indicates that using both the Keccak and modular units simultaneously can achieve a total performance improvement of approximately 88%. Since our Keccak implementation is based on an OpenCores design,

any future improvements to that core would increase the relative performance contribution of our modular unit. The performance data from [13] is included for reference only, as it does not provide exact clock cycles and uses milliseconds as the unit of measurement, making a direct comparison difficult.

Additionally, the hardware resource utilization of the proposed module is presented in Table 2. The implementation by Mao et al. combines the Keccak module with additional polynomial multiplication/addition units and an NTT/iNTT module. In comparison, our resource utilization, which includes only the Keccak and modular units, is approximately 9% lower in terms of Flip-Flops (FFs) and Look-Up Tables (LUTs). The reason for this small 9% difference, despite having fewer internal modules, is the input size of the AXI Data FIFO. To process ALTEQ's hash input, which can be up to 13 KB, we configured the AXI Data FIFO depth to 8192, leading to substantial FF and LUT consumption. Besides this, we use 6 DSPs for Keccak and 2 for the modular unit, for a total of 8, which is 2–4 more than used by Mao et al. Due to the very large input data, BRAM utilization is also high to accommodate temporary storage. This is not due to an implementation issue but is inherent to the algorithmic characteristics of ALTEQ, suggesting that further optimization strategies should be considered. Furthermore, regarding the maximum operating frequency (Fmax), our implementation was targeted for 100 MHz, whereas Mao et al. achieved 161 MHz. We anticipate that the Fmax can be improved through further pipelining enhancements in the modular unit.

A noteworthy aspect of these results is that significant additional performance improvement can be achieved with the modular unit at the cost of only about 2% more FFs, 1% more LUTs, and 2 additional DSPs. Since the resource footprint of the modular unit is small, this indicates that it provides a low-cost method for accelerating ALTEQ's performance.

## VII. DISCUSSIONS

In this section, we discuss aspects of the proposed HW/SW co-design module that require further consideration. By presenting the limitations of this study and suggesting directions for future research, we aim to provide a foundation for operating ALTEQ more securely and efficiently.

- **Algorithm and Platform Specificity:** The hardware module proposed in this study is optimized for the operational structure of ALTEQ. Since parameters such as the modulus, operands, and Keccak input length were configured according to ALTEQ's specifications, direct application or porting to other algorithms may be difficult. Furthermore, as it performs Barrett-based modular arithmetic, it is incompatible with implementations that use Montgomery or Plantard multiplication. This limitation could be addressed by designing a unified module that integrates various multiplication methods

with configurable parameters; however, we leave this for future work.

- **Lack of Side-Channel Attack Analysis:** This research focuses on accelerating the performance of ALTEQ, and we have achieved significant improvements by accelerating its Keccak and modular operations, which is designed as constant-time. However, constant-time designs on FPGAs are still vulnerable to side-channel attacks, prompting the development of further countermeasures such as masking. As ALTEQ's modular operations, in particular, use the core secret information (the ATF) as an operand, side-channel security could be achieved by analyzing the data path of this secret information and applying appropriate countermeasures.
- **Limitation of Comparison:** As this study presents the first HW/SW co-design implementation of ALTEQ, a direct comparison with other implementations of the same algorithm is difficult. Therefore, our comparison was limited to the performance of CRYSTALS-Dilithium, for which many acceleration methods have already been proposed.
- **Feasibility of Acceleration:** ALTEQ's modular operations are invoked within triple or quadruple nested for loops, resulting in a very high call frequency. The operation involves a computation between the ATF and a column, processing `vec_size` amounts of data read from each address. Therefore, adopting a method that reads and processes data in chunks that are multiples of `vec_size` could reduce data transfer time and accelerate the entire scheme. However, the module proposed in this study is parameterized to handle only a single loop's data at a time, thus consuming more clock cycles on data transfer. We leave this implementation strategy for future work as a potential method for further accelerating ALTEQ's performance.

## VIII. CONCLUSION

To explore the hardware acceleration potential of non-lattice-based algorithms, an under-researched area in post-quantum cryptography, this study proposes and implements the first HW/SW co-design for ALTEQ, a candidate from the NIST PQC additional signature competition. We began by performing software profiling, which identified the main performance bottlenecks in ALTEQ as the hash function—accounting for over 80% of the total computation—and the modular arithmetic operations invoked within nested loops. Based on this analysis, we developed a strategy to maximize overall system performance by offloading these two computationally intensive operations to an FPGA-based hardware accelerator.

To achieve this, we extended and modified an existing Keccak accelerator design to handle ALTEQ's large data inputs and designed a new, resource-efficient, pipelined unified modular unit that significantly reduces arithmetic latency. This work is pioneering in that it presents the first hardware acceleration architecture for ALTEQ. Furthermore,

we have demonstrated that even an algorithm not selected for standardization can achieve substantial performance gains through hardware optimization. This highlights the potential for such cryptographic schemes to be re-evaluated for specific application environments and could serve as a catalyst for future research into hardware implementations of a broader range of PQC algorithms. Future work could focus on enhancing security by applying side-channel attack countermeasures and further improving performance by optimizing data transfer methods.

## CONFLICT OF INTEREST

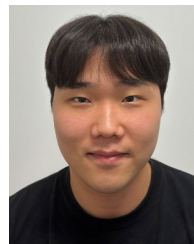
The authors have no conflict of interest.

## 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] National Institute of Standards and Technology. *Post-Quantum Cryptography Standardization Project*. Accessed: Oct. 1, 2025. [Online]. Available: <https://csrc.nist.gov/Projects/post-quantum-cryptography>
- [3] Korean Post-Quantum Cryptography Research Center. *Kpqc Competition*. Accessed: Oct. 1, 2025. [Online]. Available: [https://kpqc.or.kr/contents/03\\_exhibit/board.html?board\\_id=board\\_competition&mode=view&no=6&cate=](https://kpqc.or.kr/contents/03_exhibit/board.html?board_id=board_competition&mode=view&no=6&cate=)
- [4] P. Sanal, E. Karagoz, H. Seo, R. Azarderakhsh, and M. M. Kermani, “Kyber on ARM64: Compact implementations of kyber on 64-bit ARM cortex—A processors,” in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, vol. 2021, 2021, pp. 424–440.
- [5] D. O. C. Greconici, M. J. Kannwischer, and D. Sprengels, “Compact dilithium implementations on cortex-M3 and cortex-M4,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 1–24, Dec. 2020.
- [6] Y. Kim, J. Song, and S. C. Seo, “Accelerating falcon on ARMv8,” *IEEE Access*, vol. 10, pp. 44446–44460, 2022.
- [7] H. Kwon, H. Kim, M. Sim, W.-K. Lee, and H. Seo, “Look-up the rainbow: Table-based implementation of rainbow signature on 64-bit ARMv8 processors,” *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 5, pp. 1–19, Sep. 2023.
- [8] M.-S. Chen, T. Güneysu, M. Krausz, and J. P. Thoma, “Carry-less to bike faster,” in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, Cham, Switzerland: Springer, 2022, pp. 833–852.
- [9] S. Pircher, J. Geier, A. Zeh, and D. Mueller-Gritschneider, “Exploring the RISC-V vector extension for the classic McEliece post-quantum cryptosystem,” in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 401–407.
- [10] K.-A. Shim, S. Lee, and N. Koo, “Efficient implementations of rainbow and UOV using AVX2,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 245–269, Nov. 2021.
- [11] H. Cheng, G. Fotiadis, J. Großschädl, and P. Y. A. Ryan, “Highly vectorized SIKE for AVX-512,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 41–68, Feb. 2022.
- [12] M. Bläser, D. H. Duong, A. K. Narayanan, T. Plantard, Y. Qiao, A. Sipasseuth, and G. Tang, “The alteq signature scheme: Algorithm specifications and supporting documentation,” NIST PQC Submission, USA, Tech. Rep., 2023.
- [13] G. Mao, D. Chen, G. Li, W. Dai, A. I. Sanka, Ç. K. Koç, and R. C. C. Cheung, “High-performance and configurable SW/HW co-design of post-quantum signature CRYSTALS-dilithium,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 16, no. 3, pp. 1–28, Sep. 2023.
- [14] Y. Xing and S. Li, “A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 328–356, Feb. 2021.
- [15] S. Pendyala, R. Magesh, E. B. Kavun, and A. Aysu, “Outrunning the millennium falcon: Speed records for falcon on fpgas,” *Cryptol. ePrint Arch.*, 2025.
- [16] A. Dolmeta, S. Di Matteo, E. Valea, M. Carmona, A. Loiseau, M. Martina, and G. Masera, “TYRCA: A RISC-V tightly-coupled accelerator for code-based cryptography,” in *Proc. Design, Autom. & Test Eur. Conf. (DATE)*, Mar. 2025, pp. 1–7.
- [17] T. Fritzmann, G. Sigl, and J. Sepúlveda, “RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 239–280, Aug. 2020.
- [18] E. Alkim, H. Evkan, N. Lahr, R. Niederhagen, and R. Petri, “ISA extensions for finite field arithmetic: Accelerating kyber and NewHope on RISC-V,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 219–242, Jun. 2020.
- [19] M. Szegedy, “Spectra of quantized walks and a  $\sqrt{\delta\epsilon}$ -rule,” 2004, *arXiv:quant-ph/0401053*.
- [20] S. Tani, “An improved claw finding algorithm using quantum walk,” in *Proc. Int. Symp. Math. Found. Comput. Sci.*, 2007, pp. 536–547.
- [21] O. Goldreich, S. Micali, and A. Wigderson, “Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems,” *J. ACM*, vol. 38, no. 3, pp. 690–728, Jul. 1991.



**YONGRYEOL CHOI** (Student Member, IEEE) received the B.S. degree from the Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, Korea, in 2024, where he is currently pursuing the M.S. degree in cyber security. His research interests include optimization of cryptographic algorithms, post-quantum cryptography implementation, module validation program, and SW/HW co-design for IoT devices.



**JAESEOK LEE** (Student Member, IEEE) received the B.S. degree from the Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, South Korea, in 2025, where he is currently pursuing the M.S. degree in cyber security. His research interests include optimization of cryptographic algorithms, post-quantum cryptography implementation, and SW/HW co-design for IoT devices.



**YOUNGBEOM KIM** (Student Member, IEEE) received the B.S. degree from the Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, South Korea, in 2021, and the M.S. degree in financial information security from Kookmin University, in 2023, where he is currently pursuing the Ph.D. degree. His research interests include optimization of cryptographic algorithms, post-quantum cryptography implementation, module validation program, and SW/HW co-design for IoT devices.



**WILLY SUSILO** (Fellow, IEEE) received the B.S. degree in computer science from Universitas Surabaya, Indonesia, in 1993, the M.S. degree in computer science from the University of New Brunswick, Canada, in 1996, and the Ph.D. degree in computer science from the University of Wollongong, Australia, in 2001. He has been with the School of Computer Science and Software Engineering, University of Wollongong, since 2000, where he is currently a Distinguished Professor. He is also the Director of the Institute of Cybersecurity and Cryptology (iC2). Previously, he was the Head of the School of Computing and Information Technology, iC2, from 2015 to 2020. His research interests include cryptography, cybersecurity, network security, and computer security. He is a fellow of IET and Australian Computer Society (ACS).



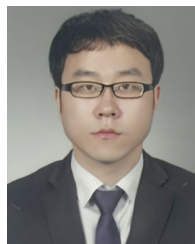
tum cryptographic module security, quantum key distribution security, and post quantum cryptography security.

**OKYEON YI** (Member, IEEE) received the Ph.D. degree in mathematics from the University of Kentucky, in 1996. From July 1999 to August 2001, he was with the Electronics and Telecommunications Research Institute, Daejeon, South Korea, as a Team Leader of mobile information security. He is currently a Professor with Kookmin University, Seoul, South Korea. His research interests include the 5G, 5G+, 6G mobile telecommunication security, drone security, quantum



hyang University. He is currently a Professor with Kookmin University, Seoul. His main research interests include internet security, authentication, access control, and formal security analysis. He is a fellow of the IET. He has served or is currently serving as a main organizer of international conferences and workshops, such as MobiWorld, MIST, SeCIHD, AsiaARES, IMIS, and so forth. He is the Editor-in-Chief of the *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*. He is on the Editorial Board for *Intelligent Automation and Soft Computing*, the *Journal of Network and Computer Applications*, the *International Journal of Ad Hoc and Ubiquitous Computing*, *Computing and Informatics*, the *Journal of High Speed Networks*, and *Security and Communication Networks*.

**ILSUN YOU** (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, South Korea, in 1997 and 2002, respectively, and the Ph.D. degree from Kyushu University, Japan, in 2012. From 1997 to 2004, he was with THINmultimedia, Inc., Internet Security Company Ltd., and Hanjo Engineering Company Ltd., as a Research Engineer. He was an Associate Professor with the Department of Information Security Engineering, Soonchun-



(SAIT) and the Samsung DMC Research and Development Center, from September 2011 to April 2014. He was a Senior Research Member with the Affiliated Institute of ETRI, South Korea, from 2014 to 2018. Currently, he is an Associate Professor with Kookmin University, South Korea. His research interests include public-key cryptography, its efficient implementations on various IT devices, cryptographic module validation program, network security, and data authentication algorithms.

**SEO G CHUNG SEO** (Member, IEEE) received the B.S. degree in information and computer engineering from Ajou University, Suwon, South Korea, in 2005, and the M.S. degree in information and communications from Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea, in 2007, and the Ph.D. degree from Korea University, Seoul, South Korea in 2011. He was a Research Staff Member with the Samsung Advanced Institute of Technology

...