A 1.96mm² Low-Latency Multi-Mode Crypto-Coprocessor for PKC-based IoT Security Protocols

Cheng-Rung Tsai, Ming-Chun Hsiao, Wen-Chung Shen, An-Yeu (Andy) Wu, and Chen-Mou Cheng

Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, TAIWAN
E-mail: {raulshepherd, mingchun, wtshen}@access.ee.ntu.edu.tw, E-mail: {andywu, doug}@ntu.edu.tw

Abstract — In this paper, we present the implementation of a multi-mode crypto-coprocessor, which can support three different public-key cryptography (PKC) engines (NTRU, TTS, Pairing) used in post-quantum and identity-based cryptosystems. The PKC-based security protocols are more energy-efficient because they usually require less communication overhead than symmetric-key-based counterparts. In this work, we propose the first-of-its-kind tri-mode PKC coprocessor for secured data transmission in Internet-of-Things (IoT) systems. For the purpose of low energy consumption, the crypto-coprocessor incorporates three design features, including 1) specialized instruction set for the multi-mode cryptosystems, 2) a highly parallel arithmetic unit for cryptographic kernel operations, and 3) a smart scheduling unit with intelligent control mechanism. By utilizing the parallel arithmetic unit, the proposed crypto-coprocessor can achieve about 50% speed up. Meanwhile, the smart scheduling unit can save up to 18% of the total latency. The crypto-coprocessor was implemented with AHB interface in TSMC 90nm CMOS technology, and the die size is only 1.96 mm². Furthermore, our chip is integrated with an ARM-based system-on-chip (SoC) platform for functional verification.

Keyword - Public-key cryptography, crypto-coprocessor, IoT, SoC

I. INTRODUCTION

Public-key cryptography (PKC) is prevailing in today’s Internet age, ranging from secured data exchange to electronic commerce. It provides a high-efficiency key management, in which the private keys need not to be transmitted nor be revealed to anyone. Furthermore, PKC can support digital signatures as an efficient means of authentication. As the applications of Internet-of-Things (IoT) systems become more popular, information security of those connected devices becomes an important issue. Due to limited power/energy constraint, deploying cryptography for IoT systems is extremely challenging. In symmetric-key cryptosystems, more bits will need to be transmitted over the air for achieving certain level of security [1]. However, the huge data/key exchange in IoT systems leads to big overhead in terms of hardware cost and energy consumption. Hence, it is advantageous to apply PKC for efficient transmission of security keys, which achieving the same security level. Researchers have demonstrated that it is possible to run PKC on sensors with acceptable performance, and there have been several attempts in employing software-based PKC to secure inter-sensor communication [2].

However, there is an emerging threat to the prevailing PKC due to the recent development of quantum computers. It has been shown in [3] that both of RSA and elliptic curve cryptography (ECC), the two most popular PKCs, could be broken by quantum computers in polynomial time. Hence, the so-called “post-quantum cryptography” [4] have been proposed to against such a catastrophic attack. Therefore, we focus on two types of post-quantum PKC in this work, including NTRU [5] and tame transformation signature (TTS) [6] in our co-processor design. Besides, in recent days, identity-based (ID-based) cryptography also becomes more important due to the advantage of free of public key infrastructure (PKI) management. Foreseeing the need of ID-based cryptography in IoT applications, the optimal ate pairing (OAP) [7] cryptosystem is also included in our design.

Our approach provides a foundation for future-proof security using processor-based PKC for general data security in IoT security protocols. We present a multi-mode crypto-coprocessor which supports three different PKCs. Firstly, we design corresponding instruction set to the aforementioned three cryptosystems (NTRU, TTS, Pairing). To enhance the energy efficiency, we proposed two design techniques:

1) An AXPY engine (multiplying vector A with vector X, plus vector Y): The AXPY engine is a specific computation unit for parallelizing crypto-based arithmetic operations. The parallel computation can reduce not only the operation time of encryption or decryption, but also the total energy consumption.

2) A smart scheduling unit (SSU): The coprocessor can achieve less latency on data transmission and scheduling by employing an intelligent control mechanism. As a result, we can increase the effective work load of the AXPY engine to achieve low latency and low energy consumption.

In our crypto-coprocessor design, the communication mechanism between the main processor and Advanced High performance Bus (AHB) are also included. It was implemented in TSMC 90nm CMOS technology, and the die size is only 1.96 mm². Furthermore, our chip is verified by an ARM-based SoC platform with ARM processor for system verification.

The rest of this paper is organized as follows. Section II describes the related cryptography algorithms. In Section III, the proposed hardware architecture and three design techniques are demonstrated, respectively. The chip implementation and system verification are discussed in Section IV. Finally, Section V draws the conclusions.
II. REVIEW OF RELATED CRYPTOGRAPHY ALGORITHMS

In this work, we aim to design and implement a multi-mode proof-of-concept PKC-based system. We choose three types of PKCs to support in our crypto-coprocessor. First, NTRU provides encryption for key exchanging. Second, TTS enjoys the benefit of executing much faster than traditional cryptosystems on the same hardware, making them ideal for securing sensors in IoT systems. Lastly, we support the identity-based cryptography which is considered to be the key feature of the security system.

NTRU is a lattice-based cryptosystem [5], whose security is based on the shortest vector problem in a lattice. The main operations of NTRU are realized in a truncated polynomial ring. Specifically, these operations involve polynomial ring multiplications (denoted as $\circ$) of two polynomials of degree N-1 whose coefficients are from integers modulo q, which is the most time-consuming operation in NTRU. The product of two polynomials $A$ and $B$

\[
A(X) = a_0 + a_1X + a_2X^2 + \cdots + a_{N-1}X^{N-1},
\]

\[
B(X) = b_0 + b_1X + b_2X^2 + \cdots + b_{N-1}X^{N-1},
\]

\[
C(X) = A(X) \circ B(X),
\]

where $C(X)$ can be calculated as:

\[
c_k = \sum_{i+j=k \mod N} a_i b_j = a_0 b_0 + a_1 b_{N-1} + \cdots + a_{N-1} b_1.
\]

In order to improve the performance of NTRU, it is necessary to develop a hardware accelerator that will speed up the polynomials ring multiplications.

Multivariate PKC is a kind of PKC whose trapdoor one-way function takes the form of a multivariate polynomial map over a finite field [8]. The security assumption relies on the NP-completeness of solving a set of multivariable quadratic polynomial equations. However, the big public keys of a multivariate PKC create a storage problem. TTS [6] is one of the key-shortening techniques. By utilizing a tame map, which is a polynomial map with relatively few terms in the equations. It can be easily invertible through solution of linear equations without a low degree explicit inverse. The main operations in the TTS involve solving a system of linear equations. There are several algorithms that can efficiently solve this problems, such as systolic Gaussian elimination and the Wiedemann algorithm [9]. Both algorithms require matrix operations, such as vector-based multiplications and additions.

Pairing is a type of PKC in which a publicly known string representing an individual or organization is used as a public key. A bilinear pairing (or simply pairing) is a map of the form:

\[
e : G_1 \times G_2 \rightarrow G_T,
\]

where $G_1, G_2$ are additive groups and $G_T$ is a multiplicative group. In this work, we will focus on implementing OAP [7] over a specific Barreto-Naehrig curve. For pairing-based scheme, finite field arithmetic plays the major computational part over all algorithms. It contains all kinds of huge number unit operation such as 256-bit finite field multiplication, upon which, direct implementation may be inefficient.

According to the three types of PKCs we discussed above, some specific crypto-based arithmetic operations are required. They usually cause the bottlenecks in the implementation. However, we note that these specific operations can be calculated by some basic operations in the single computational unit. Moreover, we will propose a low-latency and efficient coprocessor to support all of the specific operations.

III. PROPOSED CRYPTO-COPROCESSOR AND HARDWARE ARCHITECTURE

A. Overall Architecture for the Proposed Crypto-Coprocessor

The overall architecture of the proposed multi-mode crypto-coprocessor design for three different kinds of PKC algorithm is shown in the Fig. 1. It consists of five blocks: The top controller and the SSU are hierarchical control units of the system. The former maintains the ports of the coprocessor, while the later deals with the instruction decoding and data fetching for the computational unit. The AXPY engine is the main computational unit. Then, several special-designed features are included in this crypto-coprocessor.

B. Instruction Set Design

We design our instruction set corresponding to the three related algorithms of cryptography for our crypto-coprocessor. Table I shows overall instruction set of the crypto-coprocessor. Related instructions are done by SSU without unnecessary bubbles.
insertion to enhance the efficiency of the crypto-coprocessor. System-level instructions are in charge of the communication between the crypto-coprocessor and other components on the Bus. The top controller handles the system-level instructions for loading/writing and interruption operations.

**TABLE I INSTRUCTION SET OF THE CRYPTO-COPROCESSOR**

<table>
<thead>
<tr>
<th>Instruc.</th>
<th>Type</th>
<th>Meaning</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>nop</td>
<td>R</td>
<td>No operation, buffer instruction</td>
<td>single</td>
</tr>
<tr>
<td>setrn</td>
<td>R</td>
<td>Set register N</td>
<td>single</td>
</tr>
<tr>
<td>shuffle</td>
<td>R</td>
<td>Shuffle the data in register N and X by A</td>
<td>single</td>
</tr>
<tr>
<td>pol_mul</td>
<td>R</td>
<td>Polynomial ring multiplication</td>
<td>multi</td>
</tr>
<tr>
<td>add</td>
<td>R</td>
<td>q type : 16 slots parallel addition</td>
<td>single</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i type : 256-bit number addition</td>
<td>multi</td>
</tr>
<tr>
<td>sub</td>
<td>R</td>
<td>q type : 16 slots parallel substation</td>
<td>single</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i type : 256-bit number substation</td>
<td>multi</td>
</tr>
<tr>
<td>mul</td>
<td>R</td>
<td>q type : 16 slots parallel multiplication</td>
<td>single</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i type : 256-bit number multiplication</td>
<td>multi</td>
</tr>
<tr>
<td>move</td>
<td>S</td>
<td>Move data from one bank to another</td>
<td>multi</td>
</tr>
<tr>
<td>jump</td>
<td>S</td>
<td>Set the program counter to corresponding address</td>
<td>multi</td>
</tr>
<tr>
<td>branch</td>
<td>S</td>
<td>Conditionally jump according to the data loaded</td>
<td>multi</td>
</tr>
<tr>
<td>input</td>
<td>T</td>
<td>Load the data from outside memory</td>
<td>multi</td>
</tr>
<tr>
<td>output</td>
<td>T</td>
<td>Write the data to outside memory</td>
<td>single</td>
</tr>
<tr>
<td>interrupt</td>
<td>T</td>
<td>Interrupt the main processor and stay in idle</td>
<td>single</td>
</tr>
</tbody>
</table>

C. AXPY Engine for Vector-based Data Processing

The main computational unit of the crypto-coprocessor is the AXPY engine in Fig. 2. It performs the vector operation to calculate $C$:

$$C = AX + Y,$$

(4)

where $A = [a_0 \ a_1 \ldots \ a_{N-1}]$, $X = [x_0 \ x_1 \ldots \ x_{N-1}]$ and $Y = [y_0 \ y_1 \ldots \ y_{N-1}]$. In other words, the AXPY performs the vector-based multiply-accumulate (MAC) operation in our crypto-coprocessor. It is the essential operations for polynomial ring multiplications in NTRU or solving linear system equations in TTS. Moreover, our AXPY engine is far more than a general MAC unit. Since the basic unit of the related algorithm is usually polynomial or vector, we select the width of the vector as basic unit of the AXPY engine to calculate $C$ in parallel:

$$C = [c_0 \ c_1 \ldots \ c_{N-1}] \text{ where } c_i = a_i x_i + y_i.$$  

(5)

Through the parallelism, the crypto-coprocessor can reduce the overall operational cycles. Meanwhile, it can avoid data scheduling inter vector operations. The vector-based MAC leads to high throughput and low energy consumption.

Beside parallel MAC set unit, additional shift-adder set is added in our AXPY engine design. Finite field arithmetic which contains all kinds of huge number unit operation is computationally more expensive compared with the formers. As a result, finite field arithmetic reduction, such as Coursely Integrated Operand Scanning (CIOS) [10] must be adopted. These kinds of algorithms deal with this issue with the principle of “multiplication and reduction”, which means the avoidance of bit-width enlargement in the process of the multiplication. Moreover, we also observe that, in the operation of polynomial ring multiplication, continuous additions occur frequently, which can be formulated as:

$$c_i = c_{i-1} + (a_i x_i + y_i).$$  

(6)

Our AXPY engine can also operate two basic operations at the same time with an additional shift-adder set.

D. Smart Scheduling Unit (SSU)

The SSU is responsible for instruction loading/decoding and data management for AXPY engine. Meanwhile, the SSU will handle scheduling instructions, such as jump and data re-scheduling for further processing. In this way, AXPY engine can concentrate on data processing and avoid unnecessary idle state. Furthermore, two smart data scheduling mechanisms are included to enhance the efficiency. 1) To deal with data loading, general processor needs either extra cycle or specific instruction to maintain. In both cases, the operation unit will be in idle state for corresponding data loading. In cryptography algorithm, the situation is much worse because key scheduling and data exchange plays a crucial role for security guarantee. To avoid unnecessary bubble inserted to AXPY engine, data pre-fetch mechanism is applied. 2) Large amount of key exchange and key scheduling would be an important part in the cryptography. Mapping to hardware operations, we should move the key or data from one part of the memory to another one. Again, in these processes, the AXPY engine might be in idle state for waiting the corresponding data or key be ready. As a result, we put the scheduling instruction behind the multiply-cycle instructions in executing order; the SSU can deal with the scheduling instruction while the AXPY engine executes operating instruction. Due to the data pre-fetch mechanism as
proposed above, the SSU is able to detect and handle the data moving instructions during the AXPY engine processing the multi-cycle instructions.

IV. CHIP IMPLEMENTATION AND SYSTEM VERIFICATION

The multi-mode PKC co-processor was designed and fabricated in TSMC 90nm CMOS technology. The die photo of the crypto-coprocessor chip is shown in Fig.3(a). The chip contains five memory macros, and can communicate with Master processor via the AHB bus through the master port and slave port. Most importantly of all, the similar design methodology can be applied with scalable dimensions of AXPY engine. The chip summary in the Fig.3(b) shows that its die size is only 1.96mm². The clock can be up to 200MHz in post-layout simulation. However, due to the system clock limitation of the verification platform, the crypto-coprocessor is operated at 50 MHz. In addition to NTRU, TTS, and OAP cryptosystems, our chip can also support similar crypto algorithms by compiling the operations into our instruction set. For system verification, we select the MorPACK platform [11] (see Fig. 4) as the verification platform. It is a board-level SoC system with ARM processor and AHB interface for daughter modules. We can control the whole SoC platform through programming the ARM CPU. The CPU substrate board is composed of an ARM926EJ-S CPU with two 32KB SRAMs.

We also analyze the related statistic performance improvement from proposed architecture design: 1) Due to the feature of the cryptography algorithms, the AXPY engine with additional adder set can achieve about two times faster than normal mac set. However, the cost of the adder set is only 6% of the total area. 2) In SSU, data pre-fetch mechanism avoids the AXPY engine waiting the related data loaded from the data cache. Moreover, putting the data moving instruction behind the multi-cycle instruction can achieve smart data scheduling. Data pre-fetch mechanism can speed up by 18%, while smart data movement can save about 10% of total cycles. These two mechanisms are activated by SSU, the hardware cost is only 1.5% of total area. In other words, we use limited resource to achieve remarkable performance improvement.

IV. CHIP IMPLEMENTATION AND SYSTEM VERIFICATION

In this paper, we introduce the architectural design and several specialized design techniques of a multi-mode PKC-based crypto-coprocessor. The proposed design can achieve remarkable performance improvement to reduce the latency and energy consumption. Finally, we implement this crypto-coprocessor in TSMC 90nm CMOS technology. Furthermore, the chip is integrated with an SoC system board for verification.

ACKNOWLEDGMENT

This work was supported by the National Science Council, National Taiwan University and Intel Corporation under Grant NSC101-2911-I-002-001 and Grant NTU102R7501.

REFERENCES


Fig. 3. (a) Die photo of crypto-coprocessor. (D$ denotes instruction cache and D$ denotes data cache) (b) Chip summary.