Post-Quantum Cryptographic Hardware Primitives

Lake Bu, Rashmi Agrawal, Hai Cheng, and Michel A. Kinsy
Adaptive and Secure Computing Systems Laboratory
Department of Electrical and Computer Engineering, Boston University
(bulake, rashmi23, chenghai, mkinsy)@bu.edu

ABSTRACT

The development and implementation of post-quantum cryptosystems have become a pressing issue in the design of secure computing systems, as general quantum computers have become more feasible in the last two years. In this work, we introduce a set of hardware post-quantum cryptographic primitives (PCPs) consisting of four frequently used security components, i.e., public-key cryptosystem (PKC), key exchange (KEX), oblivious transfer (OT), and zero-knowledge proof (ZKP). In addition, we design a high speed polynomial multiplier to accelerate these primitives. These primitives will aid researchers and designers in constructing quantum-proof secure computing systems in the post-quantum era.

KEYWORDS

Post-quantum cryptography, public-key system, key exchange, oblivious transfer, zero-knowledge proof, FPGA-based prototyping.

1 INTRODUCTION

In the last three years, we have witnessed a raft of breakthroughs and several key milestones towards the development of general quantum computers. These advances do bring with them critical challenges to classical cryptosystems like RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography), and ElGamal. The strength of these classic algorithms rests on the hardness of integer factorization and discrete logarithm problems, which do not hold under quantum computing approaches. Thus, researchers have been actively investigating new algorithms and designs for cryptosystems for the post-quantum era. Among these techniques, designs based on Ring-learning with errors (Ring-LWE) [2] thus far have proven to be the most promising approach. Ring-LWE-based cryptosystems have the following advantages (i) their security reduction is a modification of the shortest vector problem (SVP) and closest vector problem (CVP), which are known to be NP-hard, and so far there are no efficient classical or quantum algorithms to solve them; (ii) they can support homomorphic encryption (HE) schemes; (iii) they have much smaller key size comparing with other cryptosystems; (iv) finally, in some cases, they lend themselves to more efficient hardware implementations than their classical competitors. In contrast to the extensive literature on the study and software implementation of the Ring-LWE algorithm, there has been little work on its efficient hardware implementation. Recently, a handful of works have explored the FPGA implementation of the KEX [3], and even less the PKC [4]. There is also a general lack of discussion on the design and hardware implementation of other cryptographic primitives such as oblivious transfer (OT) and zero-knowledge proof (ZKP), which play critical roles in many applications such as private machine learning and crypto-currencies using blockchain. Therefore, in this work, we construct a small representative set of reusable, standalone hardware modules of these post-quantum cryptographic primitives (PCPs). They can serve as the fundamental building blocks for a wide range of secure systems. In the work we demonstrate (1) a high speed polynomial multiplier design to aid in the efficient hardware implementation of these primitives, and (2) new algorithms for the OT and ZKP primitives.

2 THE PCP HARDWARE PRIMITIVES
2.1 The Public-Key Cryptosystem (PKC) and Key Exchange (KEX) Primitives

The detailed algorithms of the public-key cryptosystem (PKC) and key exchange (KEX) can be found in [2] and [1], respectively. For brevity, we will only briefly introduce the PKC algorithm, since many of its sub-modules are reused in the KEX primitive.

ALGORITHM 2.1. Let the ring R_q be $R_q = R/\langle q \rangle = \mathbb{Z}_q[x]/\langle f(x) \rangle$, where $f(x) = x^n + 1$ is an irreducible polynomial with n a power of 2, and $q \equiv 1 \mod 2n$ is a large prime number. Thus R_q is a ring of integer polynomials modulo both $f(x)$ and q, and it has q^n elements. Let $\mathcal{X}$ be a Gaussian distribution of "small" errors/noise. If $t = \lfloor \frac{q}{2} \rfloor$, $a, b \in R_q$ and $s, e, r_0, r_1, r_2 \leftarrow \mathcal{X}$, then the public key encryption protocol between Alice and Bob is as follows.

Key generation: Alice picks s and a random e to generate the public key $pk = \{a, b\}$ and the private key $sk = \{s\}$ by:

$$b = a \cdot s + e \tag{1}$$

Encryption: Bob converts his message (plaintext) into a binary vector m of length n, and generates the cipher $\{c_0, c_1\}$ as:

$$\begin{cases} c_0 & = b \cdot r_0 + r_2 + tm, \\ c_1 & = a \cdot r_0 + r_1. \end{cases} \tag{2}$$

Decryption: Alice decrypts the cipher by:

$$m = \lceil (c_0 - c_1 \cdot s)/t \rfloor, \tag{3}$$

where $\lceil \rfloor$ stands for taking the nearest binary integer.

The basic operations of the algorithms are: polynomial addition, polynomial subtraction, scalar multiplication, scalar division then taking the nearest binary integers, and polynomial multiplication. Most of the operations are component-wise, or can be reduced to conditional assignment. The polynomial multiplication operation has the highest hardware implementation complexity. An efficient multiplication module will substantially improve the hardware implementation efficiency of the entire hardware crypto-primitive suite. Figure 1 shows a system architecture using the commonly shared hardware modules.

One of the common implementations of the polynomial multiplier, is negative wrapped convolution combined with butterfly number-theoretic transform (NTT, the finite field version of FFT). This approach takes $O(n \log n)$ multiplications and has a time complexity of $O(\log n)$. In this work, we are introducing a new and high-speed design of the modular polynomial multiplier, named *Preemptive Adaptive Reduction Multiplier* (PARM).

Figure 1: The three core building blocks for the primitives: *Key Generation (KeyGen), Encryption (Enc),* **and** *Decryption (Dec).*

It calculates the generalized representation of the product in advance. Thus, given two polynomial multiplicands, their product can be computed as fast as in one step.

Alg. 1: Preemptive Adaptive Reduction Multiplier (PARM)

```
1   Let a = {a_0,⋯,a_{n-1}}, b = {b_0,⋯,b_{n-1}} ∈ Z_q[x]/⟨f(x)⟩
       (where f(x) = x^n + 1) be two n-bit vectors, and
       P(X) the primitive polynomial of the ring.
2   Let d = {d_0,⋯,d_{n-1}}, e = {e_0,⋯,e_{n-1}} where d_i, e_i are
       merely variable names.
3
4   Precompute:
5       ĉ ← d ⊛ e   (⊛ for convolution) such that
6       ĉ = ĉ_0 + ĉ_1 x^1 + ⋯ ĉ_{n-1} x^{n-1} + ĉ_n x^n + ⋯ ĉ_{2n-2} x^{2n-2}
7       # Approach 1: by using P(x) for reduction
8           for i=n to 2n-2 do
9               x^i = l_0 + l_1 x^1 + ⋯ + l_{n-1} x^{n-1}
10          By substituting {x^n,⋯x^{2n-2}} to ĉ
11          c = c_0 + c_1 x^1 + ⋯ c_{n-1} x^{n-1}
12      # Approach 2: by using f(x) for reduction
13          c ← ĉ/f(x) = c_0 + c_1 x^1 + ⋯ c_{n-1} x^{n-1}
14      Denote c_i = g_i(d,e), where g_i is a general
              representation of c_i by d, e.
15
16  Real-time:
17      for i=0 to n-1 do
18          c_i ← g_i(a,b)
19      end for
20
21  return c
```

As shown in the algorithm 1 outline, the bulk of the work is performed in the "Precompute" stage, which is done only once in the lifetime of the multiplier's. The real time computation is just n calculations of $c_i \leftarrow g_i(a,b)$, which can be done fully in parallel (1 cycle) given enough multipliers (n^2). The resource and time complexities for the PARM algorithm are $O(n^2)$ multiplications and $O(1)$ cycles latency, while the NTT-based complexities are $O(n\log n)$ multiplications and $O(\log n)$ cycles latency.

2.2 The Oblivious Transfer (OT) Primitive

The OT mechanism enables a receiver to choose and receive a certain piece of information out of many pieces from the sender, while remaining oblivious to the other pieces. The sender is also oblivious to the exact piece selected. The OT is a widely used protocol in privacy-preserving computations between two or multiple parties. The proposed OT primitive is constructed on the foundation of the PKC primitive. We denote $KeyGen(s,a) = b$ as the *Key Generation* module, $Enc_{pk}(m) = \{c_0, c_1\}$ as the *Encryption* function, and $Dec_{sk}(\{c_0, c_1\}) = m$ as the *Decryption* module - Figure 1. The proposed OT algorithm over ring R_q is as follows.

ALGORITHM 2.2. Suppose Alice uses $KeyGen()$ to generate and send a public key to Bob, and keeps the private pairing key to herself. Alice has l n-bit binary messages $\{m_1, \cdots, m_l\}$ and l n-bit random vectors $\{r_1, \cdots, r_l\}$.

(1) Alice sends $\{r_1, \cdots, r_l\}$ to Bob. Bob chooses the cth vector r_c in order to acquire m_c. Thus Bob generates a random binary vector $K \in R_2^n$ and computes v to send to Alice:
$$v = r_c + Enc_{pk}(K). \tag{4}$$

(2) For all $i \in \{1, 2, \cdots, l\}$, Alice computes the set $\{m_i'\}$ and sends it back to Bob:
$$m_i' = Dec_{sk}(v - r_i) \oplus m_i, \tag{5}$$
where $\oplus$ is bitwise XOR.

(3) Bob computes his desired m_c while remaining oblivious to other m_i, where $i \neq c$:
$$m_c = m_c' \oplus K. \tag{6}$$

2.3 The Zero-Knowledge Proof (ZKP) Primitive

The ZKP enables an entity to prove to a verifier that it knows a secret value s, without revealing any information (including the value of s) apart from the fact that it knows the value. Similar to the OT primitive, the ZKP primitive is designed using the building blocks the PKC algorithm in section 2.1.

ALGORITHM 2.3. Suppose Alice has a secret value s and needs to prove her ownership of s to Bob.

(1) Alice uses $KeyGen(a, s)$ to generate b. Alice selects a binary vector m, and samples $e', r \leftarrow \mathcal{X}$ to generate c:
$$c = a \cdot r + mt + e', \tag{7}$$
where $t = \lfloor \frac{q}{2} \rfloor$.
Alice sends $\{a, b, m, c\}$ to Bob, and keeps s to herself.

(2) Bob samples $u \leftarrow \mathcal{X}$, and interactively sends it to Alice.

(3) Alice responds with x to Bob:
$$x = r + s \cdot u. \tag{8}$$

(4) Bob computes and verifies if:
$$\lceil (c - a \cdot x + b \cdot u)/t \rfloor \overset{?}{=} m, \tag{9}$$
where $\lceil \rfloor$ stands for taking the nearest binary integer.
If the equality of [Eq. 9] stands, then Alice has successfully proved her ownership of s to Bob.

REFERENCES

[1] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-quantum Key Exchange-A New Hope.. In *USENIX Security Symposium*, Vol. 2016.

[2] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1–23.

[3] Tobias Oder and Tim Güneysu. 2017. Implementing the NewHope-Simple key exchange on low-cost FPGAs. *Progress in Cryptology–LATINCRYPT* 2017 (2017).

[4] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. 2014. Compact ring-LWE cryptoprocessor. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 371–391.