공학석사 학위논문

완전동형암호를 위한 RNS 기반 암호문 연산에 대한 FPGA 구현

2022년 8월

서울대학교 대학원

전기 · 정보공학부

김 지 환

완전동형암호를 위한 RNS 기반 암호문 연산에 대한 FPGA 구현

지도교수 백 윤 흥

이 논문을 공학석사 학위논문으로 제출함
2022년 8월

서울대학교 대학원
전기 · 정보공학부
김 지 환

김지환의 석사 학위논문을 인준함
2022년 8월

위 원 장 _____ 심 재 웅 (인)

부위원장 _____ 백 윤 흥 (인)

위 원 _____ 최 우 석 (인)

초 록

4차 산업 혁명의 물결과 함께 빅데이터와 AI 기술이 발전하고 활용이 증가함에 따라 컴퓨터의 데이터 처리량도 증가하고 있다. 이제 개인의 컴퓨팅 환경으로는 그만큼의 데이터 양과 연산을 감당하기 힘들어지고 있기에 아마존, 마이크로소프트와 같은 기업은 클라우드 서비스를 통해 고성능의 원격 컴퓨팅 환경을 제공하고 있다. 그런데 이러한 원격 컴퓨팅 환경은 통신 과정을 암호화할 수 있지만, 연산 과정은 복호화된 상태에서 이루어져 민감한 데이터가 손실, 유출되는 등 프라이버시 보존에 취약하다는 단점을 가지고 있다. 이를 해결하기 위해 동형암호를 통해 암호화된 상태에서 연산을 수행하는 방법이 최근 주목받고 있다. 특히, 완전동형암호는 기존 동형암호 알고리즘의 약점이었던 연산 횟수 제한을 없애고 무한히 연산을 수행할 수 있는 기술로 그 활용이 기대된다. 그러나 완전동형암호를 이용하면 연산양을 크게 증가시키기 때문에 그에 따른 성능 오버헤드가 문제점으로 지적받고 있다.

본 논문에서는 이러한 단점을 해소하기 위해 FPGA를 이용한 하드웨어 가속기 구현을 통해 암호문 연산의 성능을 개선하는 방법을 제시한다. 먼저 암호문 연산에 사용되는 모듈러 연산 모듈을 크리티컬 패스를 최소화하도록 설계하고, 모듈러 연산 모듈을 기반으로 fully pipelined된 NTT 버터플라이 유닛을 설계한다. 그리고 NTT 버터플라이 유닛이 레벨에 따라 독립적인 데이터를 병렬로 연산 수행하도록 처리하여 실행 시간을 줄이는 것이 본 연구에서 제안하는 기법이다. 제안된 모델의 성능을 측정하기 위해 완전동형암호 암호문 곱셈 연산을 FPGA에서 수행한 결과 소프트웨어 대비 성능이 향상되었음을 확인할 수 있었다.

주요어 : 완전동형암호, 프라이버시 보존, FPGA
학 번 : 2019-26912

목 차

표 목차

그림 목차

제 1 장 서론

4차 산업 혁명의 핵심인 빅데이터와 AI의 활용도가 나날이 증가하면서, 컴퓨터가 처리하는 데이터양과 학습망의 크기가 기하급수적으로 증가하고 있다. 하지만 개인 컴퓨팅 환경의 연산 능력 및 성능으로는 더 이상 그만큼의 계산과 데이터양을 감당하기 힘든 수준에 다다르고 있다. 이에 아마존, 마이크로소프트, 구글과 같은 글로벌 IT 기업들은 데이터를 효율적으로 처리하기 위한 고성능의 클라우드 서비스를 사용자에게 제공하고 있다.

그렇지만 원격 컴퓨팅 환경은 프라이버시 보존에 취약하다는 문제가 있다. 왜냐하면 클라우드 환경의 통신 과정에서는 데이터를 암호화할 수 있지만, 암호화된 데이터를 복호화한 다음 연산이 이루어지기 때문이다. 또한 사용자가 직접 연산 환경을 통제, 감시하지 못하기 때문에 민감한 데이터들의 소실, 변조, 유출은 방지하기 어려운 것이 현실이다. 이에 대응하여 K-익명성, Differential Privacy와 같은 솔루션이 등장했지만, 이들 역시 내부자 위협이나 데이터의 유출을 완전히 방지하지 못하며, 오히려 데이터 자체의 손실이 커져 연산의 정확도가 낮아지는 단점을 갖고 있다.

이러한 상황을 타개하기 위한 해결책으로 동형암호가 주목을 받고 있다. 동형암호는 데이터가 암호화된 상태로 연산을 수행할 수 있어 그 내용을 제 3자로부터 원천적으로 봉쇄할 수 있는 암호화 방법이다. 오직 사용자만이 원문을 확인할 수 있으므로 프라이버시 보존이 이루어진다.

하지만 동형암호를 적용시키면 암호문의 크기와 복잡한 연산으로 인해 성능 오버헤드가 증가하게 된다. 일례로, IBM에서 개발한 동형암호 라이브러리 HElib은 암호문 연산의 성능이 일반 연산보다 50만 배 이상 느리다. 이를 극복하기 위해 GSW, CKKS와 같은 새로운 동형암호 알고리즘이 제안되거나, CPU, GPU 등을 활용한 가속 연구가 지속되었지만, 동형암호가 널리 쓰이기에는 아직도 큰 성능 오버헤드를 가지고 있다.

이에 따라 하드웨어를 이용해 동형암호를 가속하려는 연구가 진행되고 있다 [7], [8], [9]. 이 논문들은 완전동형암호인 CKKS 알고리즘을 FPGA 또는 ASIC으로 구현하여 암호문 연산의 성능을 향상시켰지만, 완전동형암호의 특징인 암호문 연산 횟수의 제한을

없애기 위한 부트스트래핑 과정에 대한 가속은 아직 미비하다 [7], [8]. [9]의 경우 부트스트래핑 과정의 가속이 구현되어 있지만, non-packed 데이터에 대한 것으로 한정되어 있어 온전히 구현한 것으로 보기 힘들다.

본 논문에서는 완전동형암호의 암호문 연산을 FPGA로 가속하기 위한 연구를 진행했다. FPGA는 프로그래밍 가능한 장점을 갖고 있어 프라이버시를 유지한 채 다양한 연산을 수행할 수 있는 응용 방안을 구현하는 데 있어 중요한 역할을 할 것으로 예상된다.

제 2 장 배경 지식

본 장에서는 이 연구에 사용된 완전동형암호인 RNS-CKKS 스킴에 대한 배경 지식을 간략히 설명한다.

제 1 절 완전동형암호

동형암호는 데이터가 암호화된 상태에서 덧셈과 곱셈을 계산할 수 있는 암호 알고리즘이다. 암호문을 이용한 연산의 결과가 복호화된 평문의 연산 결과와 동일하다는 의미에서 동형암호라 불린다 [1]. 이러한 특성으로 인해 동형암호로 암호화된 데이터를 연산하는 데에는 비밀키가 필요하지 않게 된다. 동형암호의 덧셈, 곱셈 보존은 다음과 같은 수식으로 표현된다.

$$E(m_1) + E(m_2) = E(m_1 + m_2)$$
$$E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2)$$

또한, 동형암호는 암호화된 상태에서 덧셈과 곱셈을 수행할 수 있어 컴퓨터가 처리하는 모든 연산을 표현할 수 있기 때문에 튜링 완정성을 갖는다.

그런데 동형암호는 연산을 할 수록 잡음이 커져 평문이 훼손되기 때문에 특정 횟수만큼의 연산만 수행할 수 있다. 이러한 연산 횟수의 제한을 해소하기 위해 제안된 알고리즘이 바로 완전동형암호이다. 완전동형암호는 암호화된 상태의 데이터를 연산할 수 있으면서 그 연산을 무한히 수행 가능하다. 연산을 수행할 때마다 쌓이는 잡음을 줄이기 위해, 완전동형암호 알고리즘은 특정한 횟수의 연산을 수행할 때마다 부트스트래핑 과정을 필요로 한다 [3]. 부트스트래핑 과정을 거치면 동일한 평문에 대한 암호문을 잡음이 작은 암호문으로 대체한다.

제 2 절 RNS-CKSS

CKKS는 동형암호 스킴의 일종으로, 고정소수점을 이용한 실수의 근사연산을 지원한다 [2]. RNS-CKKS 알고리즘은 기존의 CKKS 스킴을 RNS(Residue Number System) 기반으로 표현한다 [4]. RNS는 큰 수를 그보다 작은 여러 소수들에 의한 나머지의 집합으로 표현한다. Chinese Remainder Theorem(CRT)을 이용하면 이렇게 나뉘어진 나머지에서 원래 값을 복원 가능하다. 이러한 특징을 이용하면 원래 값에서 도출된 나머지 값들은 독립적으로 연산이 가능하기 때문에, 큰 값의 연산을 피하고 작은 값을 병렬적으로 연산하는 것이 가능해진다. RNS-CKKS를 이용하면 기존의 CKKS 스킴보다 더 빠른 속도로 연산이 가능하며, 필요한 메모리의 크기도 줄어들게 된다.

제 3 장 설계 및 구현

본 장에서는 완전동형암호 가속기의 설계와 구현에 대하여 상세히 설명하고자 한다. RNS-CKKS 알고리즘은 모듈러 연산이 기본이 되므로, 먼저 모듈러 연산 모듈의 설계에 대해 설명한다. 그 다음, 모듈러 연산 모듈을 기반으로 하는 NTT 버터플라이 유닛의 구현에 대해 설명할 것이다. 마지막으로 FPGA 동작 시 데이터의 흐름과 메모리 구조에 대해 설명하도록 한다.

제 1 절 모듈러 연산 모듈

RNS-CKKS 스킴의 특성에 따라, 암호문 연산 과정에서 수행하는 계산은 모두 모듈러 연산이다. 따라서 암호문 연산의 기반이 되는 모듈러 연산 모듈을 먼저 설계했다. 타겟 FPGA 보드에서 250Mhz의 주파수로 동작하기 위해 모듈러 덧셈, 모듈러 뺄셈 모듈은 critical path가 최소화되도록 설계했다. 그림 1은 각각 모듈러 덧셈, 모듈러 뺄셈 모듈을 나타낸 것이다.

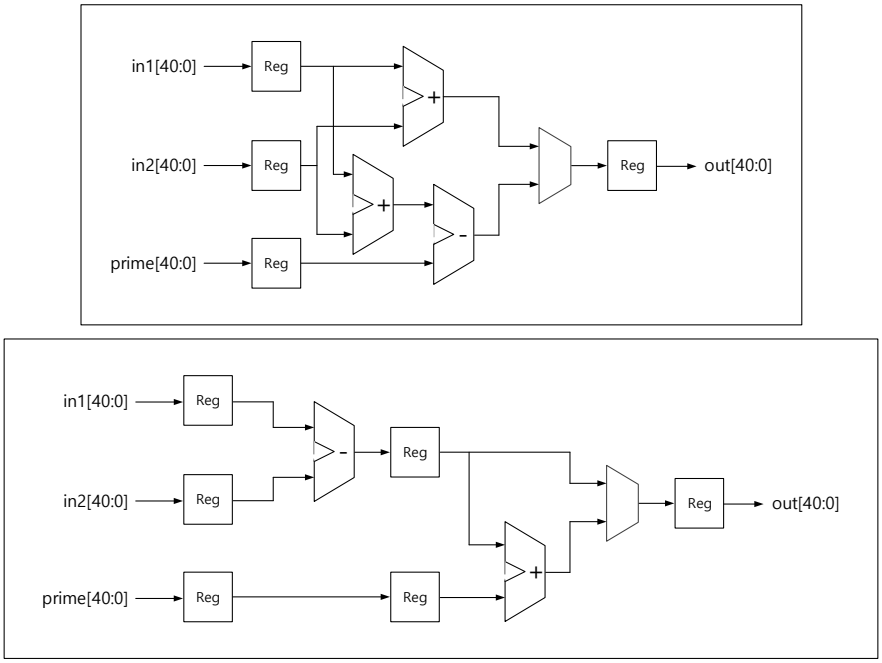

그림 1 모듈러 덧셈, 뺄셈 모듈

이렇게 설계된 모듈러 덧셈, 모듈러 뺄셈 모듈은 각각 1, 2 사이클만에 연산 결과가 출력된다.

모듈러 곱셈 모듈은 Barrett Reduction 알고리즘을 적용해 모듈을 설계했다. 일반적인 모듈러 곱셈 연산은 곱셈 연산을 먼저 수행한 다음 모듈러 연산을 위해 나눗셈기가 필요하다. 하지만 Barrett Reduction 알고리즘을 이용하면 나눗셈 연산을 곱셈 연산 과정으로 변환해 나눗셈기를 사용하지 않아 연산이 복잡해지는 것을 방지할 수 있다 [6]. 이 때, 곱셈 연산 과정은 Full integer multiplier, Half integer multiplier 모듈의 연산을 수행한다.

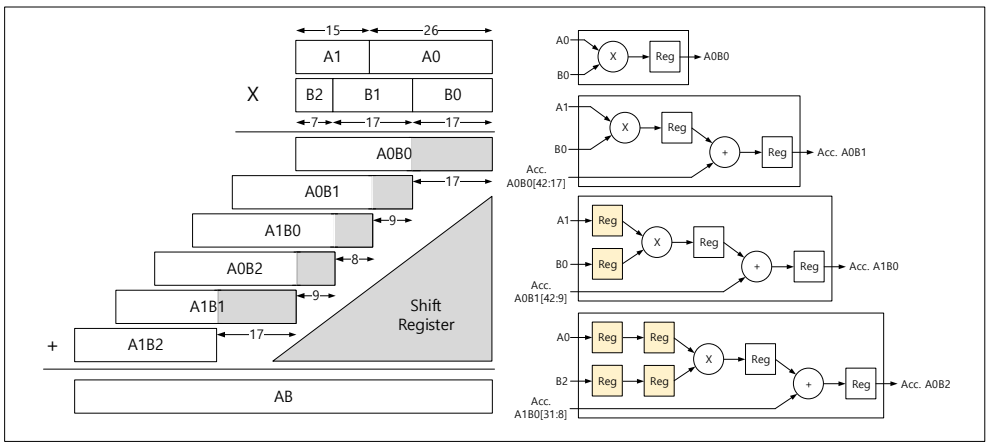

그림 2 Full integer multiplier

그림 2는 Barrett Reduction을 위한 Full integer multiplier 모듈을 나타낸 그림이다. FPGA에서 사용되는 자원을 줄이기 위해, 해당 모듈을 설계할 때 DSP48 slice와 DSP 내부의 레지스터를 활용했다.

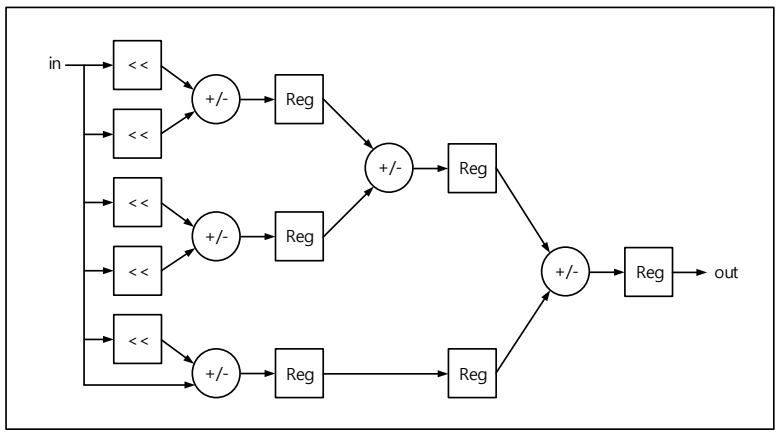

그림 3 Half integer multiplier

그림 3은 Half integer multiplier 모듈을 나타낸 것이다. 이 연산의 모듈러스(modulus)는 전부 2의 제곱의 합으로 표현이 가능한 Solinas Prime이기 때문에, 곱셈 연산을 덧셈과 시프트 연산으로 대체해 설계했다 [6]. Barrett Reduction을 적용한 모듈러 곱셈 모듈은 Full integer multiplier와 두 번의 Half integer multiplier 연산을 거쳐 총 20 사이클만에 연산 결과가 출력된다.

제 2 절 NTT 모듈

앞서 설명한 모듈러 연산 모듈을 기반으로 fully-pipelined된 NTT 버터플라이 유닛을 설계했다. NTT 연산은 prime값에 따라 병렬적으로 수행할 있으므로, 그림 4와 같이 하나의 NTT 모듈에 하나의 버터플라이 유닛만 구성하여 데이터 접근에 대한 복잡성을 줄였다.

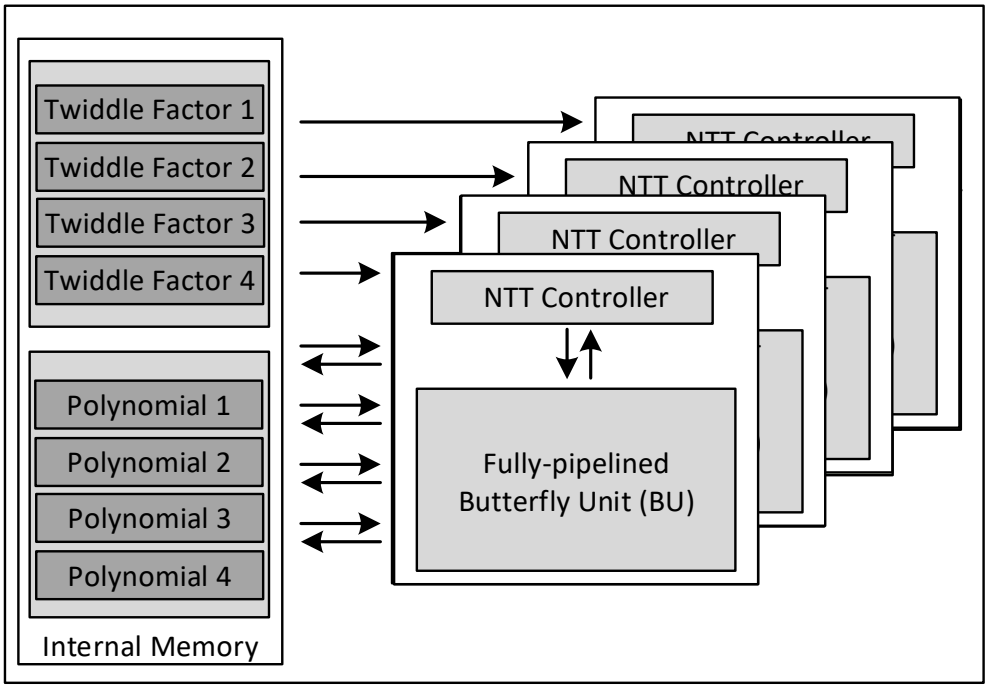

그림 4 NTT 모듈

제 3 절 FPGA 메모리 구조

그림 5는 FPGA에 구현된 가속기가 암호문 연산을 수행 중일 때의 메모리 구조를 나타낸 것이다. 암호문 연산은 FPGA 내에서 전부 이루어지지만, 연산 초기에 CPU를 통해 암호문과 키를 넘겨받아야 연산을 수행할 수 있다. CPU에서 HEAAN 라이브러리를 통해 암호문과 키를 생성하면, 이 데이터를 PCIe를 통해 FPGA의 DRAM으로 우선 전송한다. 이 중에서 크기가 큰 암호키는 HBM(High Bandwidth Memory)에 저장하고, 암호문은 연산에 자주 불러오기 때문에 내부 메모리(SRAM)에 저장한다.

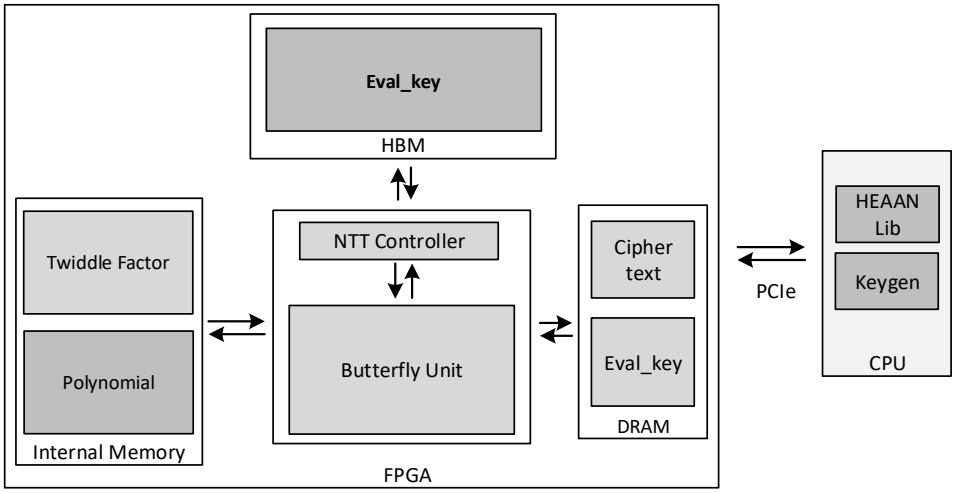

그림 5 CPU-FPGA 메모리 구조

제 4 장 실험 결과

본 장에서는 앞 장에서의 설계를 바탕으로 완전동형암호 가속기를 구현하여 성능을 측정했다. 완전동형암호 알고리즘으로는 RNS-CKKS HEAAN 라이브러리를 이용했으며, Xilinx Alveo U280 FPGA 보드에서 250MHz로 구동시켰다. 비교를 위해 먼저 CPU에서 동작시킨 결과와 성능을 비교하고, 이전 연구와 본 연구의 성능을 비교했다. 이 때 CPU에서의 실행 환경은 Intel(R) Core(TM) i7-4790을 3.60GHz에서 동작시켰다.

	CPU	Proposed Model	Speedup
실행 시간	355.2 ms	19.97 ms	**x 17.8**

표 1 암호문 곱셈 실행 시간 비교

	Chen [10]	Kim [5]	Proposed Model
실행 시간	109 ms	3.76 ms	**1.97 ms**

표 2 INTT 연산 실행 시간 비교

표 1은 암호문 곱셈 연산을 CPU에서 실행한 결과와 FPGA에서 동작시킨 결과를 나타낸 표이다. 실험 결과 FPGA에서 SW 대비 약 17.8배의 성능을 향상시킨 것이 나타났다. 또한 제안하는 모델의 throughput을 측정했을 때는 6.16 ms마다 한 번씩 암호문 곱셈 연산이 실행되고, 초당 약 162회 연산을 수행하는 것으로 나타났다.

표 2는 INTT 연산의 실행시간을 이전 연구의 성능과 비교한 결과를 나타낸 것이다. Chen [10] 대비해서는 약 55.3배, Kim [5]와 비교했을 때는 약 1.9배 향상된 성능을 보였다.

표 3은 위 실험에 사용된 제안하는 모델의 FPGA 자원 사용량을 나타낸 것으로, LUT, FF, DSP 세 가지 자원 모두 FPGA의 전체 자원 중 30% 이하만을 사용했다.

LUT (%)	FF (%)	DSP (%)
237372 (18.2%)	148652 (5.7%)	2368 (26.2%)

표 3 FPGA 자원 사용량

제 5 장 결론

본 논문에서는 FPGA를 활용한 완전동형암호 암호문 연산에 대한 가속기 모델을 제시하고 있다. 완전동형암호 연산을 위한 모듈을 설계하고 FPGA 내부 메모리 등을 활용하면서, 완전동형암호 활용의 가장 큰 문제인 성능 오버헤드를 해결하기 위한 방법을 제안했다. 제안하는 모델의 실험 결과, CPU와 이전 연구 대비 성능이 향상된 것을 확인할 수 있었다. 또한 현재 설계한 모델은 완전동형암호 연산의 FPGA 구현에 초점을 두고 연구를 진행했기 때문에 가속시킬 수 있는 여지가 남아있다. 실제로 표 3에서 나타난 것과 같이 FPGA에서 사용하지 않은 자원이 존재하므로, 남아있는 자원을 활용하면 더 많은 연산 모듈을 통해 성능을 향상시킬 수 있을 것이다. 본 연구가 완전동형암호의 암호문 연산뿐만 아니라 부트스트래핑을 가속하는 연구에 활용될 수 있을 것으로 기대된다.

참고 문헌

[1] Rivest, Ronald L., Len Adleman, and Michael L. Dertouzos. "On data banks and privacy homomorphisms." Foundations of secure computation 4.11 (1978): 169-180.

[2] Cheon, Jung Hee, et al. "Homomorphic encryption for arithmetic of approximate numbers." International conference on the theory and application of cryptology and information security. Springer, Cham, 2017.

[3] Cheon, Jung Hee, et al. "Bootstrapping for approximate homomorphic encryption." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2018.

[4] Cheon, Jung Hee, et al. "A full RNS variant of approximate homomorphic encryption." International Conference on Selected Areas in Cryptography. Springer, Cham, 2018.

[5] Kim, Sunwoong, et al. "Hardware architecture of a number theoretic transform for a bootstrappable RNS-based homomorphic encryption scheme." 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2020.

[6] Kim, Sunwoong, et al. "FPGA-based accelerators of fully pipelined modular multipliers for homomorphic encryption." 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, 2019.

[7] Riazi, M. Sadegh, et al. "HEAX: An architecture for computing on encrypted data." Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. 2020.

[8] Han, Mingqin, et al. "coxHE: A software-hardware co-design framework for FPGA acceleration of homomorphic computation." 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2022.

[9] Samardzic, Nikola, et al. "F1: A fast and programmable accelerator for fully homomorphic encryption." MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture. 2021.

[10] Chen, Donald Donglong, et al. "High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems." IEEE Transactions on Circuits and Systems I: Regular Papers 62.1 (2014): 157-166.

Abstract

FPGA Implementation of RNS-based Ciphertext Arithmetic Operation for Fully Homomorphic Encryption

Jeehwan Kim
Electrical and Computer Engineering
The Graduate School
Seoul National University

As Big Data and AI, the key technologies of the 4th industrial revolution, are developed and widely used, the amount of data processed is growing. Because it is difficult to handle the amount of data and calculation in personal computing environment, companies such as Amazon and Microsoft provide high-performance remote computing environment through cloud services. Communication in the remote computing environment is processed with encrypted data, however, computation is performed using decrypted data, so it has the disadvantage of being vulnerable to privacy preservation, such as loss or leakage of sensitive data. To solve this problem, recent studies take note of Homomorphic Encryption which is a method of performing calculation in encrypted data. In particular, Fully Homomorphic Encryption is a technology that able to perform operation infinitely, it is expected to be used widely by removing the constraint on the number of operations, which is a weakness of Homomorphic Encryption scheme. However, the performance

overhead is pointed out as a problem because Fully Homomorphic Encryption increases amount of computation significantly.

In this paper, we propose a method to improve the performance of ciphertext arithmetic operation by implementing a hardware accelerator on FPGA. First, modular arithmetic module used for ciphertext operation is designed to minimize the critical path, and a fully-pipelined NTT butterfly unit is designed based on the modular arithmetic module. And the NTT butterfly units process independent data by level to perform parallel processing to reduce the execution time. We evaluate performance of the proposed model and it demonstrates performance improvement in ciphertext multiplication.

Keywords : Fully Homomorphic Encryption, Privacy-Preserving, FPGA
Student Number : 2019-26912