A 55nm 50nJ/encode 13nJ/decode Homomorphic Encryption Crypto-Engine for IoT Nodes to Enable Secure Computation on Encrypted Data

Insik Yoon, Ningyuan Cao, Anvesha Amaravati, Arijit Raychowdhury
Georgia Institute of Technology, Atlanta, Georgia

Abstract—We present a 55nm test-chip prototype of a crypto-system (encryption and decryption) implementing Homomorphic Encryption that can enable computation on encrypted data. Test-chip measurements show 50nJ/encode and 13nJ/decode thus making the cryptosystem suitable for sensor-nodes and IoT applications.

I. INTRODUCTION

With the widespread proliferation of internet of things (IoT) sensors and devices, the role of cloud based computing is becoming ever important. This is associated with increasing concerns of privacy; where leaks can happen in communication (between IoT nodes and the cloud), storage (both in the node and the cloud) and computation (in the cloud). While both communication and storage can be secured with conventional cryptosystems; the fact that computation in the cloud is done primarily on unencrypted data exposes it to vulnerabilities. Users need to either (1) share their key with the cloud service provider, which is a security concern (Fig.1), or (2) download the data on their device to compute, which the IoT node may not computationally support. One way to preserve confidentiality of data when outsourcing computation is to enable computation on the encrypted data itself [1], a paradigm that has been revolutionized by recent advances in homomorphic encryption (HE) which allows arbitrary operations (say, function f) on ciphertexts [2]. HE is a public-key system based on Learning with Errors (LWE). Further, fully HE is quantum-resistant and hence expected to bring in the next paradigm of data-security. However, it is computationally and energy-expensive; requires extensive data movement, logic operations, and unique compute blocks (e.g., number theoretic transforms (NTTs)). To provide a pathway towards practical HE for low-power and secure IoT nodes (e.g., medical devices, secure financial and ID data nodes etc.) we present a 55nm hardware cryptosystem (encoder and decoder) operating at a measured FMAX of 60MHz and peak energy-efficiency of 50nJ/encode and 13nJ/decode. In prior research homomorphic encryption has been implemented on CPUs/GPUs and on FPGAs level [3] [4]. [5] presents an ASIC implementation of a number theoretic transform, which is a computational kernel used in post-quantum cryptosystems. In this paper we present a hardware macro for an end-to-end HE cryptosystem. We believe that this is the first full hardware demonstration of a HE cryptosystem and the associated system architecture and circuit macros.

II. SYSTEM AND CIRCUIT ARCHITECTURE

The proposed system implements the BGV scheme [6] [7] (Fig.2) using a 34bit prime number (q) and can encrypt 4b of plaintext per round. Hence the encrypted data is in ring $\mathbb{Z}_q / \Phi_5(X)$, where $\Phi_5(X)$ represents the 5th cyclotomic polynomial and $Z$ is the integer space. Data-invariant parameters are stored in an embedded array and read in parallel for each compute module. The datapath widths of the sub-blocks in the system are shown in the corresponding figures. The cryptosystem comprises of three key dedicated compute blocks, namely the Bluestein inverse/Number Theoretic Transform [bNTT/ibNTT], modular multiplication [MulMod] and modular add/subtraction [Add/Sub Mod]. Hardware acceleration for large integer arithmetic blocks (156b add/sub/mul) are further supported in the decryption engine. For potential integration in IoT nodes, we focus on a smaller design area by serializing the operations through efficient hardware reuse. A control block synchronizes data movement, whereas full-scan and input/output FIFOs enable debug and IP integration.
Fig. 3: Circuit diagram of the (a) key generation (b) encryption (c) decryption engine

Fig. 4: Algorithm flow-chart chart of the (a) key generation and (b) encryption engine

Fig. 5: Timing diagram of the key generation engine.

Fig. 6: Timing diagram of the encryption engine.

Fig. 7: Algorithm flow-chart chart of the decryption engine.

Fig. 3(a) illustrates the in-situ generation of the secret-key [sk] and the public-key system [pk0, pk1]. Based on a random number (input from external TRNG source), sk is generated via bNNT. The public key is generated from sk. While pk1 is generated from a TRNG (outside the chip) pk0 is generated as pk0 = 2pk0NNT*sk, where pk0NNT denotes the bNNT of a Gaussian random number (external input). The algorithm flow for keygen is explained in detail in Fig. 4(a). From Fig. 5, the total key generation requires 7172 clock cycles.

Fig. 3(b) shows the encryption scheme with input errors (parameter of LWE), plaintext [ptxt] and public keys generate ciphertext [ctxt0, ctxt1]. From Fig. 4(b), bNNT is executed first on pre-determined errors [e0, e1] and ptxt. Ctxt1 is generated by computing rNNT*pk1 + 2e1NNT and ctxt0 is generated by computing ptxtNNT + rNNT*pk0 + 2e1NNT. This is shown in the adjoining flow-chart. The timing diagram in Fig. 6 shows a total of 14294 clock cycles.

Fig. 3(c) and Fig. 7 illustrate how decryption is performed with sk and the ciphertext. The message [msg] is calculated by taking ibNNT of ctxt0 + ctxt1*sk. The msg is compared with...
0.5q and if the msg >0.5q then msg is the final message, or else its bitwise inversion is the final message. Plain decryption and decryption of addition on ciphertexts the decryption circuit receives four 34bit inputs based on different value of q [q0,q1,q2,q3]. With each q, we multiply msg with rin (i.e., the inverse of prod). prodI is the product of all q[prod]/[current q]. For each q, it accumulates the results [sum] and the results/q[bnd]. After accumulating for all 4 qs, the circuit subtracts prod for bnd times and the lsb of the computation is the result of the multiplication. This is illustrated in the adjoining flow-chart and the timing diagram in Fig.8.

One of the key modules in both the encryption and decryption paths is the bNTT/ibNTT. Fig.9 describes the bNTT architecture and the NTT serial unit. As opposed to previous implementations [5], which can perform NTT on powers of 2 only, we implement Bluestein NTT which can compute NTT on arbitrary number of points. Bluestein NTT enables multiple post-quantum cryptographic protocols to be implemented. To compute Bluestein NTT, the input is first multiplied with wh, the list of [10th primitive root of unity] ^2. Then 16-point NTTs are executed on the result with a twiddle factor w, which is the 16th primitive root of unity. 16-point NTT is implemented serially with one butterfly unit and intermediate buffers as shown in Fig 9(b). By taking 32 repetitions (8 reps for 1 round x 4) of the butterfly unit, the 16-point NTT is performed. Then the result of NTT is multiplied with rb, which is the NTT of multiplicative inverse of wh. Since the system implements a hardware version of 8 input MulMods, two 8 point MulMod in series are performed to calculate a 16-point MulMod. With the result, inverse NTT is performed with winv, the multiplicative inverse of w. Finally the MulMod of the 4th element to the 11th element of the result with ifftw, (multiplicative inverse of 16) * wh is performed to compute the final result. In the last step, Z5 * th (1st, 2nd, 3rd and 4th) outputs are read as final outputs. For ibNTT, the same operations are performed with the multiplicative inverses of wh and w.

Fig. 8: Timing diagram of the decryption engine.

Fig. 9: Circuit diagram of (a) Bluestein Number Theoretic Transform(bNTT) and (b) 16-points Number Theoretic Transform(NTT) used in bNTT

Fig. 10: Chip micrograph and characteristics
III. MEASUREMENT RESULTS

The cryptosystem is designed and fabricated in a 55nm GF CMOS process. The chip characteristics and the die-photo are shown in Fig.10. It occupies an area of 1.25mm X 2.5mm, which includes peripheral circuits, debug and test infrastructure and input/output FIFO. To enable high energy-efficiency and near-VT to sub-VT operation, the digital cells do not use stacks of more than 3-transistors and the embedded latches and flip-flops are fully interruptible. Measurements are carried out on packaged parts with high speed test-interfaces. Fig. 11(a) show a wide dynamic range of operation from 1.2 V (nominal V<sub>CC</sub>) to 280mV (deep sub-VT). A peak system power of 4.8mW decreases to only 85nW at V<sub>MIN</sub>=280mV. The wide dynamic range enables a similar wide throughput range for the encryption and decryption engines as illustrated in Fig. 11(b), thus enabling integration with IoT nodes with variable data-rates.

Fig. 12 illustrates the energy-efficiency of encoding and decoding. We observe that the system shows higher energy-efficiency (lower energy/op) as the V<sub>CC</sub> is scaled. A peak efficiency is observed at V<sub>CC</sub>=0.425V (V<sub>CC</sub>=0.4V) for the encoder (decoder) where 50.62nJ/op (13.11nJ/op) is measured. Table I shows a comparison of the proposed design with the state-of-the-art. Unlike previous contributions which were focused primarily on the implementation of the NTT, the current design presents an end-to-end cryptosystem. Measurements from the current design shows 92.4%, 87.5% improvement in power and energy/NTT and is well suited for edge-devices. Thus, a practical HE accelerator operating at high energy-efficiency is demonstrated for integration with IoT nodes, to enable secure end-to-end computing on encrypted data.

IV. CONCLUSION

A fully HE test-chip in 55nm CMOS is presented. It enables secure computation on encrypted data at 50nJ/encode and 13nJ/decode.

V. ACKNOWLEDGEMENT

This project was supported by Intel.

REFERENCES