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 energetically 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 F_{MAX} of 60MHz and peak energy-efficiency of 50nJ/encode and 13nJ/decode. In prior research homormorphic 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 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 (136b add/sub/mult) 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.

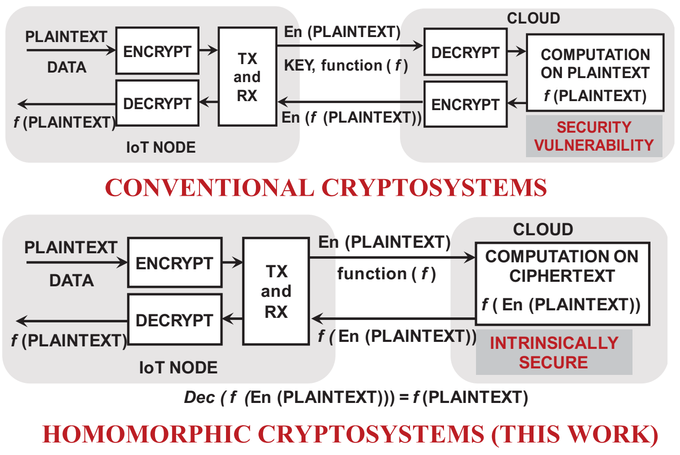


Fig. 1: Comparison between conventional crypto-system and homomorphic crypto-system

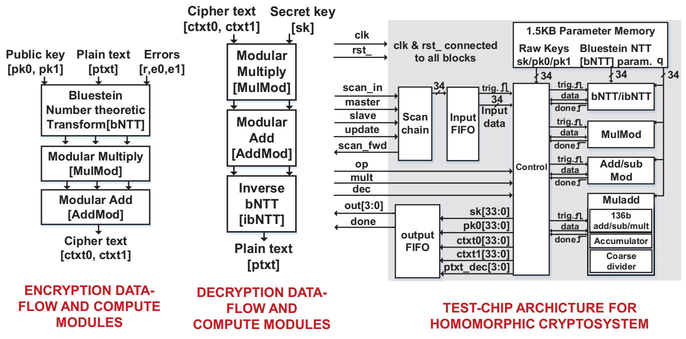


Fig. 2: Compute blocks and system architecture for the Homomorphic cryptosystem employing BGV algorithm

* 978-1-5386-9395-7/19/$31.00 ©2019 IEEE

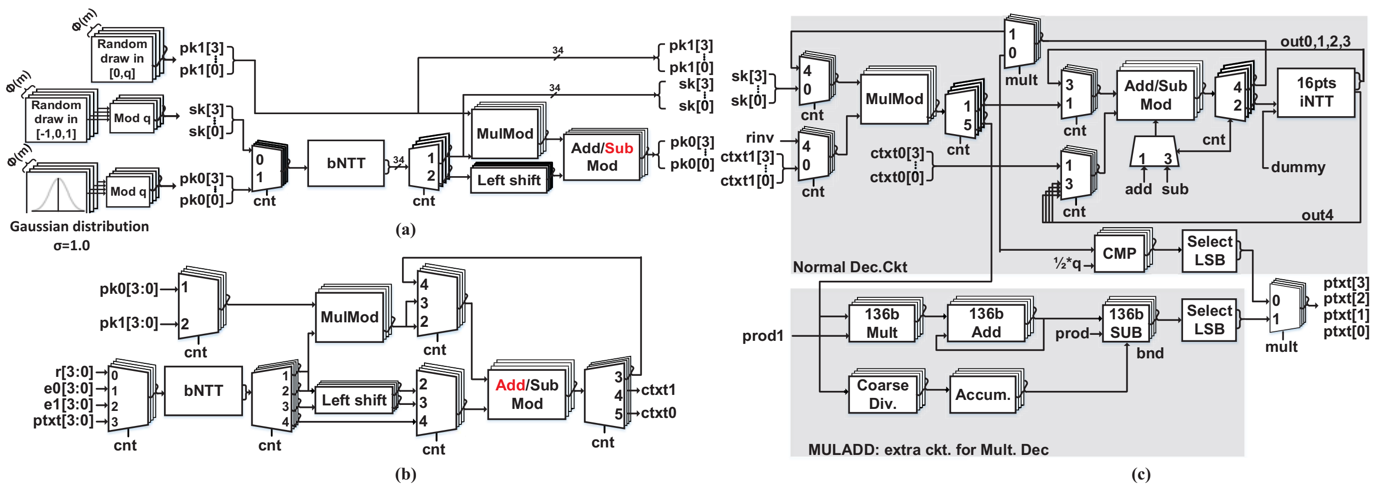
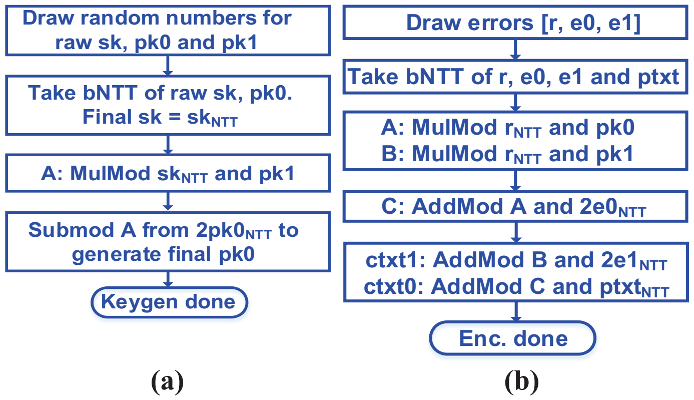


Fig. 3: Circuit diagram of the (a) key generation (b) encryption (c) decryption engine



Draw random numbers for raw sk, pk0 and pk1
Take bNTT of raw sk, pk0. Final sk = sk_NTT
A: MulMod sk_NTT and pk1
Submod A from 2pk0_NTT to generate final pk0
Keygen done
(a)
Draw errors [r, e0, e1]
Take bNTT of r, e0, e1 and ptxt
A: MulMod r_NTT and pk0
B: MulMod r_NTT and pk1
C: AddMod A and 2e0_NTT
ctxt1: AddMod B and 2e1_NTT
ctxt0: AddMod C and ptxt_NTT
Enc. done
(b)

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

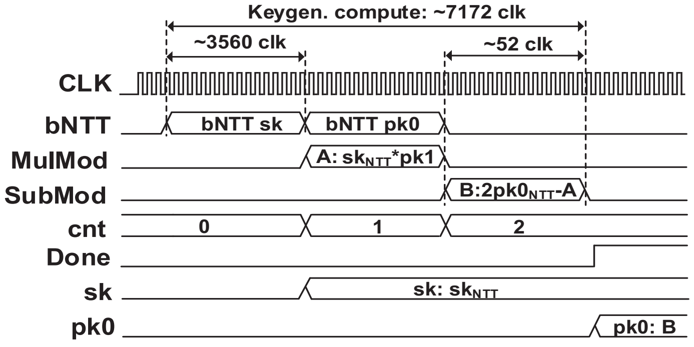


Fig. 5: Timing diagram of the key generation engine.

Fig. 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 *bNTT*. The public key is generated from *sk*. While *pk1* is generated from a TRNG (outside the chip) *pk0* is generated as $pk0 = 2pk0_{NTT}\text{-}sk{*}pk1$, where pk0_{NTT} denotes the *bNTT* 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

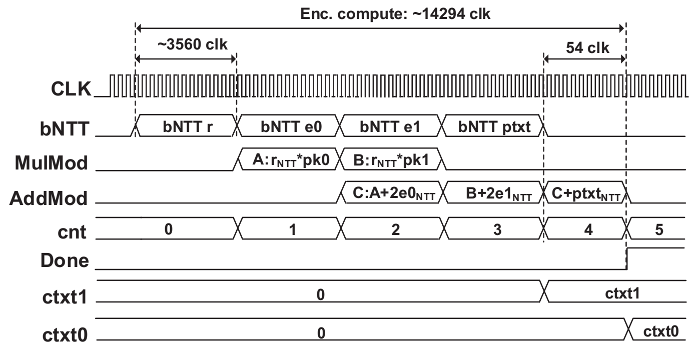
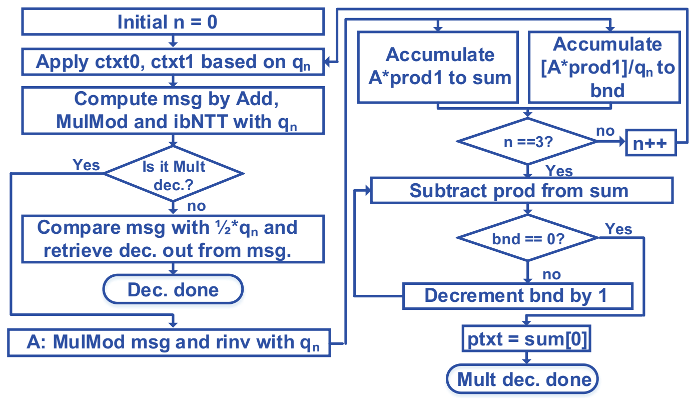


Fig. 6: Timing diagram of the encryption engine.

(parameter of LWE), plaintext [*ptxt*] and public keys generate ciphertext [*ctxt0, ctxt1*]. From Fig.4(b), *bNTT* is executed first on pre-determined errors [*r,e0,e1*] and *ptxt*. *Ctxt1* is generated by computing $r_{NTT}{*}pk1{+}2e1_{NTT}$ and *ctxt0* is generated by computing $ptxt_{NTT}{+}r_{NTT}{*}pk0{+}2e1_{NTT}$. 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 *ibNTT* of *ctxt0+ctxt1*sk*. The *msg* is compared with



Initial n = 0
Apply ctxt0, ctxt1 based on q_n
Compute msg by Add, MulMod and ibNTT with q_n
Is it Mult dec.?
Yes
no
Compare msg with ½*q_n and retrieve dec. out from msg.
Dec. done
A: MulMod msg and rinv with q_n
Accumulate A*prod1 to sum
Accumulate [A*prod1]/q_n to bnd
n ==3?
no
n++
Yes
Subtract prod from sum
bnd == 0?
Yes
no
Decrement bnd by 1
ptxt = sum[0]
Mult dec. done

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

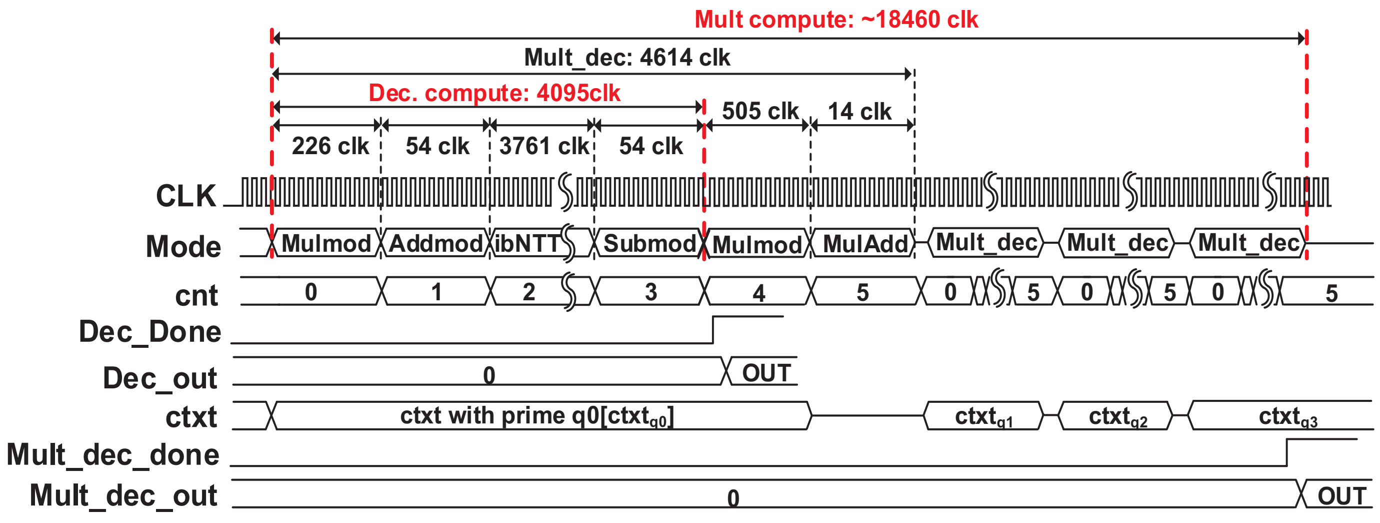
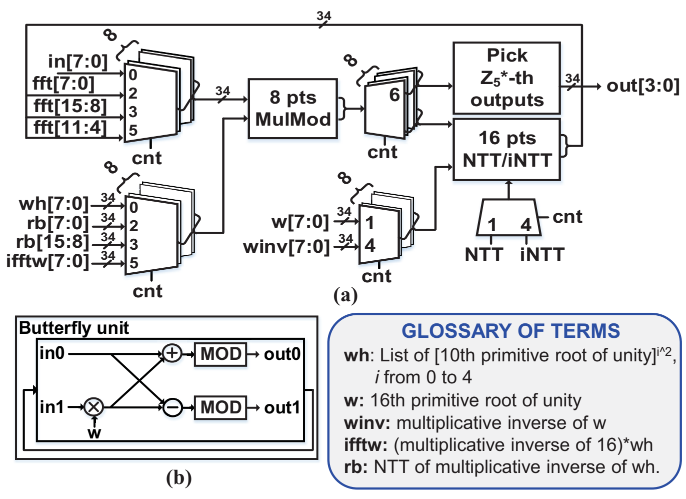


Fig. 8: Timing diagram of the decryption engine.

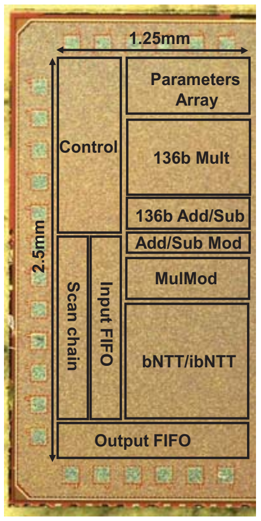


wh: List of [10th primitive root of unity]$^{i\text{^}2}$, *i* from 0 to 4
w: 16th primitive root of unity
winv: multiplicative inverse of w
ifftw: (multiplicative inverse of 16)*wh
rb: NTT of multiplicative inverse of wh.

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

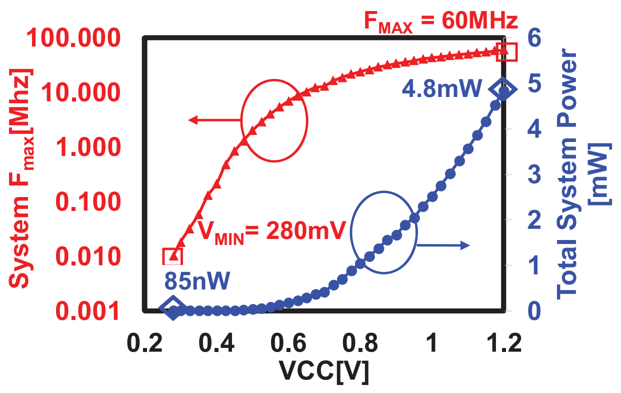
0.5q and if the *msg >0.5q* then *msg* is the final message, or else its bitwise inversion is the final message. Plain decryption and decryption of addition on ciphertext generates the correct plaintext. In order to decrypt multiplication of ciphertexts the decryption circuit receives four 34bit inputs based on different value of *q* [q_0,q_1,q_2,q_3]. With each *q*, we multiply *msg* with *rinv* (i.e., the inverse of *prod1*). *prod1* 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 *q*s, 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] i^2. Then 16-point *NTT*s 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 *MulMod*s, 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, $Z_5{}^*$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*.

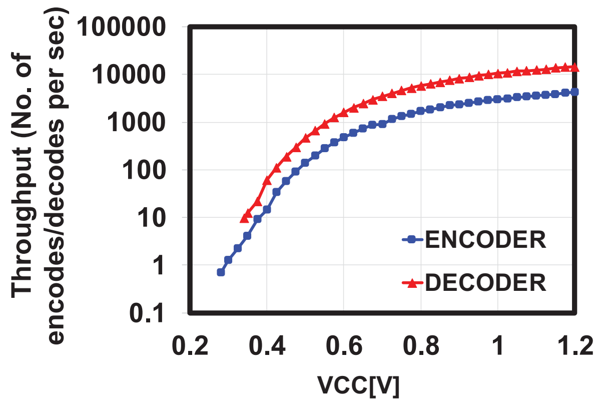


Chip Characteristics	
Technology	55nm 1P8M GF CMOS
Die Size	1.25mm x 2.5mm
Test Interface	QFN Package
ENC. Scheme	BGV
q_0	FA000001 [HEX]
q_1	EB000001 [HEX]
q_2	348000001 [HEX]
q_3	334000001 [HEX]
Cyclotomic polynomial	$\Phi_5(x) = x^4 + x^3 + x^2 + x + 1$
ENC. Throughput	4268 enc/s @ 60 Mhz
DEC. Throughput	14896 dec/s @ 60 Mhz
# plaintext bits/Enc.	4
No. of Pads	24
Scan Interface	Boundary Scan
Operating freq.	10khz - 60 Mhz
Vcc	0.28V – 1.2V

Fig. 10: Chip micrograph and characteristics



(a)



(b)

Fig. 11: (a) Measured F_{max} and total system power (b) Throughput of the encryption and decryption engine w.r.t. supply voltage

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_{CC}) to 280mV (deep sub-VT). A peak system power of 4.8mW decreases to only 85nW at V_{MIN}=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_{CC} is scaled. A peak efficiency is observed at V_{CC}=0.425V (V_{CC}=0.4V) for the encoder (decoder) where 50.63nJ/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.

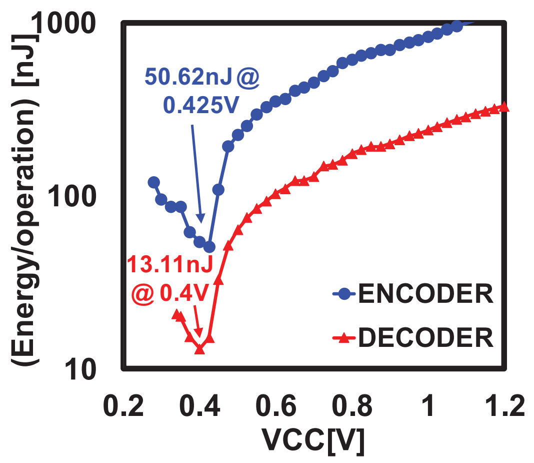


Fig. 12: Energy per encryption and decryption as a function of supply voltage

	[8]	[5]	This work
platform	Cortex-M4	40nm CMOS	55nm CMOS
q	12289		16575889409
distribution	Binomial Φ_{16}		Binomial Φ_5
NTT type	NTT		Bluestein NTT
N	512		16
Frequency (MHz)	168	300	60
No. of NTT cycles	87223	492	3560
No. of INTT cycles	97789	572	3761
NTT power(mW)	72	58.9	4.51
NTT energy(nJ/NTT)	37400	96	12.03
Cryptosystem power(mW)	NA	NA	4.8
Energy-efficiency (nJ/op)	NA	NA	50.62 (encryption)

TABLE I: Comparison with the state-of-the-art designs

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

[1] I. Verbauwhede et al., "24.1 circuit challenges from cryptography," in *ISSCC*, 2015.
[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," STOC, 2009.
[3] D. et al. Y., "Accelerating fully homomorphic encryption in hardware," *IEEE Transactions on Computers*, 2015.
[4] E. Öztürk et al., "Accelerating somewhat homomorphic evaluation using fpgas," *IACR Cryptology ePrint Archive*, 2015.
[5] S. Song et al., "Leia: A 2.05mm2140mw lattice encryption instruction accelerator in 40nm cmos," in *CICC*, 2018.
[6] Z. Brakerski et al., "(leveled) fully homomorphic encryption without bootstrapping," ITCS '12, 2012.
[7] S. Halevi and V. Shoup, "Algorithms in helib," 2014.
[8] E. Alkim et al. in *SPACE*, 2016.