

PASTA on Edge: Cryptoprocessor for Hybrid Homomorphic Encryption

Aikata Aikata¹, Daniel Sanz Sobrino^{1,2,†}, Sujoy Sinha Roy¹

¹Graz University of Technology, Austria, ² Universidad Politécnica de Madrid, Spain
{aikata,sujoy.sinharoy}@iaik.tugraz.at, daniel.sanz.sobrino@alumnos.upm.es

Abstract—Fully Homomorphic Encryption (FHE) enables privacy-preserving computation but imposes significant computational and communication overhead on the client for the public-key encryption. To alleviate this burden, previous works have introduced the Hybrid Homomorphic Encryption (HHE) paradigm, which combines symmetric encryption with homomorphic decryption to enhance performance for the FHE client. While early HHE schemes focused on binary data, modern versions now support integer prime fields, improving their efficiency for practical applications such as secure machine learning.

Despite several HHE schemes proposed in the literature, there has been no comprehensive study evaluating their performance or area advantages over FHE for encryption tasks. This paper addresses this gap by presenting the first implementation of an HHE scheme—PASTA. It is a symmetric encryption scheme over integers designed to facilitate fast client encryption and homomorphic symmetric decryption on the server. We provide performance results for both FPGA and ASIC platforms, including a RISC-V System-on-Chip (SoC) implementation on a low-end 130nm ASIC technology, which achieves a 43–171× speedup compared to a CPU. Additionally, on high-end 7nm and 28nm ASIC platforms, our design demonstrates a 97× speedup over prior public-key client accelerators for FHE. We have made our design public and benchmarked an application to support future research.

Index Terms—HHE, RISC-V, Hardware Implementation, FPGA

I. INTRODUCTION

Data breaches have raised serious privacy concerns among clients using computational or storage services offered by the service providers. While the advent of Fully Homomorphic Encryption (FHE) [1] has fulfilled the need for a privacy-preserving cryptographic protocol, it faces several challenges before practical realization. The FHE schemes transform plaintext data into much larger polynomials when homomorphically encrypted, resulting in substantial communication and computational overhead, often ranging from 10,000× to 100,000× [2]. Generating and transmitting these large ciphertexts is very expensive for the clients due to limited processing capabilities.

To address these, ‘Hybrid’ Homomorphic Encryption (HHE) was introduced [3], preventing expensive data expansion. It essentially involves using Symmetric key-based Encryption (SE) methods that can be decrypted in a homomorphic manner. The data flow from HHE to FHE is illustrated in Fig. 1. Initially, HHE schemes were designed to work with binary data

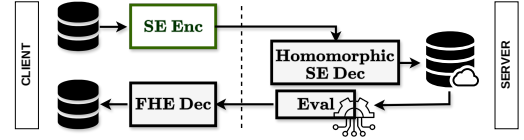


Fig. 1. Workflow of HHE schemes is illustrated here. Symmetrically encrypted ciphertexts are sent to the server. The server performs a ‘homomorphic HHE decryption’ and obtains a homomorphically encrypted ciphertext.

(\mathbb{Z}_2), such as RASTA [4], Flip [5], Filip [6], Kreyvium [7], and AgRasta [4]. However, in response to the growing need for better performance and application support, especially in integer FHE, they have evolved into schemes like MASTA [8], PASTA [9], HERA [10], and RUBATO [11] that can encrypt messages in prime fields \mathbb{F}_p , enabling operations on integers.

A. Research Gap

■ **Difference with Traditional SE.** Modern SE primitives used within the HHE framework are significantly different from traditional SE primitives such as AES [12] or ASCON [13]. The primary difference is the data type. While the traditional SE schemes are defined over boolean values (\mathbb{Z}_2), new HHE enabling SE schemes are defined over integers (\mathbb{F}_p). Thus, the data path increases significantly based on the size of modulus p . This also affects the operations/functions computed on the data. Traditional SE schemes use boolean AND, OR, and XOR operations, which are very cheap compared to integer multiplication, addition, and subtraction required by HHE enabling SE.

The SE schemes have two layers, substitution (S-box) and permutation, which provide necessary confusion and diffusion. While the traditional schemes can use a look-up-table-based S-box, this is not feasible for the new schemes due to the sheer size of the data (32–128 state elements each of size 17–60 bits). Similarly, permutation in traditional schemes involves shifting rows or columns and mixing within columns where state sizes are small (e.g., 128 bits). The permutation in new SE schemes requires expensive invertible matrix generation and multiplication of size 32×32 – 128×128 . Furthermore, while the traditional schemes have standalone descriptions, the new SE schemes utilize another expensive cryptographic standard—SHAKE128 (based on Keccak) as a pseudo-random number generator. Thus, they are bound to consume significantly more area and time than traditional SE schemes.

■ **The Challenge.** While the above subsection brings forth an important discussion regarding the new HHE enabling SE

This work was supported in part by CONFIDENTIAL-6G EU project (Grant No: 101096435) and the State Government of Styria, Austria – Department Zukunftslands Steiermark.

[†] During this work, Daniel was an Erasmus student at TU Graz.

schemes being more expensive than traditional SE schemes, it does not give an idea of the challenge of implementing them and the expected results compared to existing public-key encryption (PKE) client-side routines. In most prior PKE client-side acceleration works, the polynomial degree is $N = 2^{13}$, and for polynomial multiplication (the *most expensive* routine), a Number Theoretic Transform (NTT) based technique is utilized. Each NTT transform requires $\frac{N \cdot \log N}{2}$ multiplications. The number of such transformations required is three per modulus, which is done for three different moduli. Thus, the total number of multiplications required is $\approx 2^{19}$.

For the HHE enabling SE scheme- PASTA, there are two variants- PASTA-3 and PASTA-4. PASTA-3 the state comprises 256 elements, split into two parts of 128 each. The number of rounds is three, and each round involves a 128×128 invertible matrix generation and multiplication. The multiplicative complexity of both operations is the same- $128 \cdot 128$, and eight such operations are required. This brings the total multiplication cost to 2^{18} . It is interesting to note that, while FHE encrypts 2^{12} elements in 2^{13} degree polynomials, PASTA-3 can only encrypt $128 = 2^7$. Therefore, although PASTA-3 encrypts 32 elements twice as fast, it will need 2^6 more encryptions to encrypt 2^{12} elements, resulting in $32\times$ slower computation for data-intensive applications.

Furthermore, HHE, unlike FHE, requires SHAKE128, a giant building block even in Post-Quantum schemes [14]. Thus, while the new SE schemes offer an advantage in terms of $6\times$ lower communication, any performance or area consumption advantage is not apparent. Given that communication overhead plays a crucial role, HHE requires a thorough design exploration to offer accelerated computation and determine if the communication advantage justifies potential trade-offs.

■ **The Implementation Gap.** To the best of our knowledge, an HHE scheme has not yet been implemented in hardware. While several accelerators exist for the server [15]–[17], there are very few addressing the needs of the FHE client. The prior works on FHE PKE accelerators [18]–[22] can mitigate computation costs, but communication overhead remains a major bottleneck in practical use. Additionally, the platforms these accelerators target (e.g., AlveoU250) are prohibitively expensive for typical clients. As a result, clients face both *computational* and *communication* overheads. This makes it essential to explore how efficiently HHE schemes, which prevent communication overhead, can be implemented on resource-constrained hardware or as System-on-Chip (SoC).

B. Our Contribution

This work addresses the aforementioned research gap and proposes a hardware accelerator for an HHE scheme- PASTA. This is the first hardware realization of any HHE scheme. As a system supporting FHE client, it reduces the communication overhead due to its use of HHE and offers several orders better performance and energy efficiency than software and prior client-side PKE accelerators. The primary design technique includes the optimal realization of the invertible matrix generation and multiplication step. The design further features

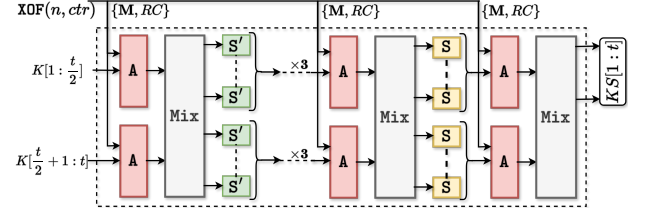


Fig. 2. The PASTA-t permutation (π) takes as input the key (K), nonce (n), and counter (ctr), and generates the truncated result- key stream (KS).

efficient resource sharing amongst all the routines. The ASIC results show a speedup of $97\times$ compared to prior PKE client accelerators. This design is further realized as a RISC-V SoC on a low-end 130nm ASIC technology and delivers up to $171\times$ speedup compared to CPU. We report a $857\text{-}3,439\times$ reduction in clock cycles compared to [9] on CPU. Furthermore, we make our design publicly available¹ to support further research.

II. BACKGROUND

Notations: We denote numbers as small letters (e.g., x), vectors of numbers as capital letters (e.g., X), and matrices as bold and capital letters (e.g., \mathbf{M}). The subscripts X_i are used for indexing coefficients in matrix (or elements in vector) or names (X_L, V_4), and the superscripts X^i are used for indexing rows in a matrix.

A. Hybrid Homomorphic Encryption

- The client uses the FHE public key to encrypt the SE private key- K homomorphically and sends it to the server at the beginning. After this, clients use HHE Encryption to encrypt the message blocks m_i using K and send the resultant ciphertext to the server whenever they need to store or process data on the cloud.
- The server evaluates the HHE decryption circuit homomorphically. This results in a homomorphically encrypted ciphertext, which the server can process.
- The client can retrieve the homomorphic ciphertext and decrypt the result using FHE secret key.

B. HHE Scheme Design Overview

In the context of HHE, the variables can be categorized into two types: public and private. As illustrated in Fig. 2, the nonce (n) and the counter (ctr) are considered public data, as they are known to both the client and the server. In contrast, the key (K) is private and exclusively known to the client. For HHD, this key is encrypted and securely transmitted to the server (done only once initially). Hence, both K and the client's message (m_i) remain concealed from the server.

With the public and private variables clearly defined, we now describe the PASTA HHE scheme. We have chosen PASTA [9] for our implementation as other HHE scheme designs can be viewed as adaptations or variations of PASTA. This scheme operates as a stream cipher and comprises two variants: 3-round PASTA-3 and 4-round PASTA-4. Fig. 2 demonstrates the PASTA permutation (π). It is important to note that the operations

¹The design and its RISC-V integration are available at https://github.com/aikata10/Pasta_RISCV

outside the square box, denoted as XOF (extendable-output function), are public. SHAKE128 is used for this. Contrarily, the operations within the box are considered private (key-dependent) and involve either addition or multiplication using modular arithmetic in \mathbb{Z}_p . Here, p can be any prime between 16 and 60 bits depending on the specific requirements of the underlying FHE scheme.

The state size ($2t$) varies between the PASTA-3 and PASTA-4 variants of the scheme, where t is the size of one block. Specifically, for PASTA-3, $2t = 128$ coefficients, while for PASTA-4, it is 64. These $2t$ coefficients are divided into two halves, X_L and X_R , and then processed via permutation. The resulting KS (KeyStream) is added to the plaintext block (size t) for encryption. Each PASTA round consists of several layers, described as follows:

- A_i (Affine Layer): For this layer, an *invertible* matrix \mathbf{M}_i and a round constant vector RC_i are generated using the SHAKE128 XOF output. Then, the layer performs $\mathbf{M}_i \cdot X_i + RC_i$ operation, where X_i represents the input state comprising t coefficients.
- Mix (Mixing Layer): Following A_i , the two halves of the state are mixed using the Mix operation. This operation transforms the state into $(2 \cdot X_L + X_R, 2 \cdot X_R + X_L)$. This step is crucial for spreading values evenly across the two-state halves.
- S'/S (S-Box Layer): The next layer involves the S-Box operation. For the final round, the cube S-Box (S) is applied, while for all previous rounds, a Feistel S-Box (S') is utilized. Both these S-boxes are *invertible*.

$$S'(X^j) = \begin{cases} X^j \pmod{p} & \text{if } j = 0, \\ X^j + (X^{j-1})^2 \pmod{p} & \text{otherwise,} \end{cases}$$

- Truncation Layer ($Trunc$): It returns the X_L state as the final output. This is applied at the end (pre-final) and truncates the output to prevent round inversion.

C. Invertible Matrix Generation Method

The affine layer in PASTA requires invertible matrices \mathbf{M}_i to ensure one-to-one mapping between input and output. The steps for generating them are as follows.

- Generate a vector of t elements using SHAKE128 XOF. This constitutes the first row of the matrix $\mathbf{M}_i^0 = [\alpha_0, \alpha_1, \dots, \alpha_{t-1}]$.
- Next, use this row to generate the remaining rows using (1) [23], [24]. This ensures that the matrix is invertible

$$\mathbf{M}_i^{j+1} = \mathbf{M}_i^j \cdot \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \alpha_0 & \alpha_1 & \alpha_2 & \dots & \alpha_{t-1} \end{bmatrix} \quad \forall 0 \leq j < t \quad (1)$$

III. DESIGN METHODOLOGY

In [9], the authors have reported the clock cycle count for PASTA encryption (also stated in Tab. II). They further give a small breakdown showing that the affine generation alone consumes 54-60% of the total clock cycles. Thus, apart

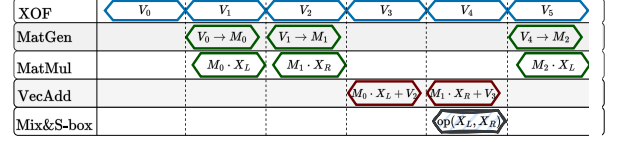


Fig. 3. The schedule for one round of the PASTA- π permutation.

from the invertible matrix generation and matrix multiplication, the pseudo-random number generation plays a significant role in determining the overall performance. With this important observation from the scheme designers of PASTA, we explore techniques for designing a highly parallel, high-performance pipelined design for modulus p in the 17-54 bits range. We do not perform round-unrolling so that the design can fit in small FPGAs (e.g., Artix-7), which are more suitable for clients.

A. XOF and Rejection Sampling Unit

A significant challenge arises in the efficient generation of pseudo-random data required by the PASTA-3/4 cryptographic schemes, which demand 2048/640 coefficients. After consuming the nonce and counter, the XOF unit generates one 64-bit coefficient per clock cycle, followed by a rejection sampling step. This process continues until all 21 pseudo-random coefficients are utilized. At this point, the XOF performs another 24-round Keccak permutation and generates the next set of 21 coefficients. This cycle repeats until the necessary amount of data is produced. A key challenge is that rejection sampling can result in up to half of the generated data being discarded, placing strain on the throughput.

To address this, we implemented a high-performance SHAKE128 design [14] that tackles this bottleneck by executing Keccak permutations in parallel with the squeeze operation, nearly doubling performance. However, this enhancement requires two 1600-bit state buffers, increasing resource usage. Further, the rejection sampling procedure for matrix and round constant vector generation is identical, necessitating an efficient integration with the XOF routine. We addressed this by incorporating a DataGen unit, which operates in a ping-pong configuration: while one vector is used to generate the matrix, the other stores XOF results for the subsequent computation. This optimization, as illustrated in Fig. 4, enables continuous data flow and improves overall throughput.

B. Operation Schedule

To ensure high performance, the XOF output must be consumed as soon as it is generated. The rate for the rejection sampling utilized can be any value between $1-2\times$; thus, an XOF data vector can be ready in at least t clock cycles. Each round requires the generation of four random vectors for the affine transformation: the first two are used for matrix generation, while the latter two serve as round constants for addition. Matrix generation is followed by matrix multiplication, while Mix and S-box operations follow the round constant addition. The challenge lies in ensuring full parallelization—matrix multiplications and Mix/S-box operations must be completed in time for the next round of XOF data to become available.

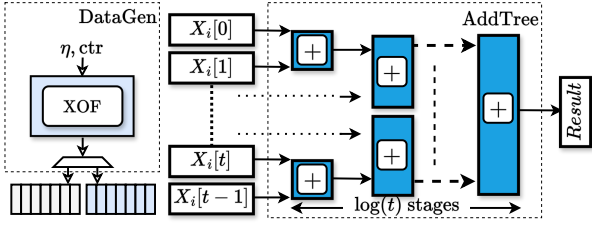


Fig. 4. The Data Generation and Adder-Tree units utilized in Matrix Generation and Multiplication. The log operation has base 2.

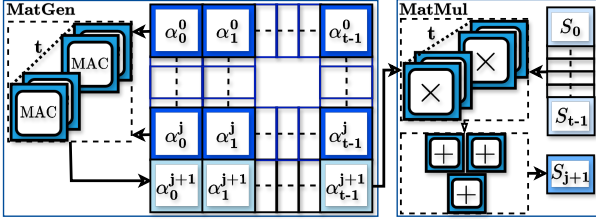


Fig. 5. The Invertible Matrix Generation and Multiplication Units.

Careful coordination is needed to prevent bottlenecks and maintain continuous data flow between rounds.

The overall flow, illustrated in Fig. 3, shows how the random vectors generated by the XOF (denoted as V_i) are either fed into the MatGen or VecAdd. Matrix multiplications are concurrent to matrix generation, while Mix and S-Box operations follow vector addition. This design ensures the on-time completion of each computation before the next round of data is generated, enabling a balance between parallelism and throughput.

C. Invertible Matrix Generation & Multiplication Unit Design

Matrix generation and multiplication introduce a computational challenge, as they require $2t^2$ modular multiplications. To finish both tasks within the stipulated time of t clock cycles, we instantiate two multiplication sets, each consisting of t modular multipliers. One set is dedicated to invertible matrix generation. As shown in (1), matrix generation also requires addition. Therefore, the unit instantiated for MatGen is a MAC (Multiply-and-Accumulate) unit, as illustrated in Fig. 5.

The matrix generation unit generates the matrix row-wise. Therefore, the multiplication is also done row-wise using MatMul unit, shown in Fig. 5. The state vector is multiplied with a row for matrix multiplication, and all the resultant elements are accumulated, similar to a vector dot-product. The multiplication is done via the second set of t multipliers. We use a pipelined adder-tree unit for additions, illustrated in Fig. 4. The time consumed for matrix generation and multiplication is $6 + t + \log_2 t$ clock cycles, accounting for the additional cycles needed to support the pipelined data flow between the MAC and matrix generation/multiplication stages.

This technique not only achieves high performance but also addresses another significant challenge—memory efficiency. By eliminating the need to store large 32×32 or 128×128 element matrices, our approach significantly reduces the memory footprint. For matrix generation, we only store the minimal

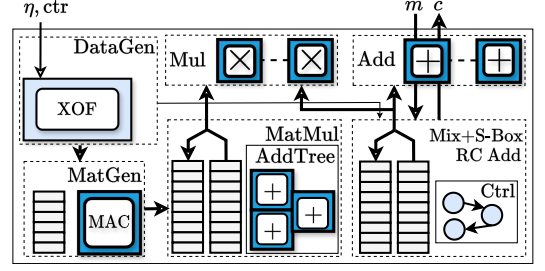


Fig. 6. The overall architecture of the hardware design for PASTA.

set of two required rows (α^0, α^j) to generate subsequent (α^{j+1}) ones, as highlighted in Fig. 5. This is due to the fact that the resultant (α^{j+1}) is immediately consumed for matrix multiplication, resulting in substantial area savings in terms of memory without compromising the throughput.

D. Vector Addition, Mix, and S-Box Unit Design

Similar to multiplication units, we employ t vector addition units. The number of vector additions required per state is t . Therefore, this unit barely consumes three clock cycles, wondering pipelining. The reason for instantiating it in a fast mode becomes evident when considering the fact that Vector Addition (RC Add), Mix, and S-Box operations need to finish in parallel with the generation of V_4 in Fig. 3. The Mix operation requires the computation of $(2 \cdot X_L + X_R, 2 \cdot X_R + X_L)$. We compute this solely using additions. Thus, three additions are required: (i) $X_R + X_L$, (ii) $X_R + (X_R + X_L)$, and (iii) $X_L + (X_R + X_L)$. The addition unit used for vector addition is also utilized for these additions, thus ensuring resource sharing.

Following this, the S-Box operations present an additional challenge. PASTA employs two S-Boxes: the cube S-Box (S), which requires two multiplications, and the Feistel S-Box (S'), which involves one multiplication and one addition. As a result, each round must compute four additions and two multiplications (or five additions and one multiplication) in parallel with the generation of t elements. The same multiplication (Mul) and addition (Add) units used in the affine transformation (MatMul, RC Add) are reused for these operations, increasing resource sharing. Since addition units are relatively low-cost, t instantiations avoid the overhead of pipelined intermediate storage using additional flip-flops.

All the multiplication and addition units operate modulo a prime- p . We observe that the moduli chosen by the authors in [9] have a Mersenne structure (e.g., 17-bit prime 65,537), enabling the use of an add-shift-based modular reduction unit following each multiplication. This technique is also utilized by prior FHE acceleration works [16].

E. Overall Architecture Design

The overall architecture of the design is shown in Fig. 6. The user provides a nonce (η), counter (ctr), and message block m , with the output being the ciphertext c . The ciphertext c is produced by performing a modular addition between the input message and the keystream (KS), which is the result of the PASTA- π permutation. The DataGen, Modular

TABLE I
THE IMPLEMENTATION RESULTS OF PASTA-3/4 ON THE ARTIX-7 BOARD
TARGETING 75MHZ CLOCK FREQUENCY.

Scheme	ω^\dagger	LUT		FF		DSP	
PASTA-3	17	65,468	49%	36,275	13%	256	35%
	17	23,736	18%	11,132	4%	64	9%
PASTA-4	33	42,330	32%	20,783	8%	256	35%
	54	67,324	50%	32,711	12%	576	78%

\dagger ' ω ' denotes the size of the Mersenne moduli.

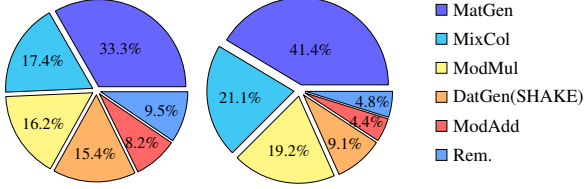


Fig. 7. Module-wise area utilization. The first pie chart shows the distribution for FPGA and the second for ASIC platforms. Rem. implies Remaining Area.

Multiplier (Mul), and Adder (Add) units are instantiated within a wrapper, enabling their reuse by both the MatMul and RC Add/Mix+S-box units for efficient resource utilization.

IV. RESULTS

A. Evaluation Platforms and Area

We choose three evaluation platforms to analyze the design for varying use cases comprehensively.

❶ **FPGA.** We set our target as the low-cost Artix-7 AC701 (xc7a200tbg676-2) FPGA at 75MHz. This FPGA features 134k Look-Up-Tables (LUTs), 269k Flip-Flops (FFs), 740 Digital Signal Processors (DSPs), and 365 Block Random Access Memories (BRAMs) with 36kb capacity. We used Xilinx Vivado 2022.2 to run synthesis, place & route. FPGA Area results are reported in Tab. I, and the area breakdown is reported in Fig. 7. The design does not require any BRAM or URAM-based memory storage. Therefore, it is not reported in the table. For PASTA-4, we place the design for a bit-width up to 54-bit to perform a comprehensive analysis.

❷ **ASIC.** We also present ASIC-synthesis results using TSMC 28nm and ASAP7 [25] 7nm libraries to compare with prior ASIC works [19], [20]. The synthesis is performed using Cadence Genus 2019.11, and clock-gating is employed to help reduce power consumption with clock frequency target as 1GHz. It reports an area consumption of 0.24 mm² and 0.03 mm² on the 28nm and 7nm processes. The maximum power consumed by the design is 1.2W. For a bit-width of 33 and 54, the area increases by $\approx 2.1\times$ and $\approx 4.3\times$.

❸ **RISC-V.** A peripheral has been developed to integrate the PASTA encryption module into the 32-bit RISC-V [26] architecture. It is connected to the central Ibex core data bus using a loosely coupled design. This peripheral is connected to the RAM via an extra bus where the peripheral acts as a master, allowing direct read access to obtain the plaintexts to be processed and subsequently encrypted. The RISC-V SoC targets 100MHz on older 130nm and 65nm technology nodes and requires 1.8 mm² (4.6 mm² with Ibex core). The peripheral processes data block-by-block, allowing a flexible

TABLE II
THE PERFORMANCE RESULTS FOR PASTA-3/4 ENCRYPTION OF ONE BLOCK
ALONG WITH THE RESULTS FOR CPU REPORTED IN PASTA.

Scheme	Elements Processed	clock cycles	FPGA (μ s)	ASIC (μ s)	RISC-V (μ s)
PASTA-3 [9]	128	17,041,380 [†]	-	-	-
PASTA-3	128	4,955	66.1	4.96	45.5
PASTA-4 [9]	32	1,363,339 [†]	-	-	-
PASTA-4	32	1,591	21.2	1.59	15.9

[†] The results are reported on an Intel Xeon E5-2699 v4 CPU [9].

number of blocks to be encrypted. This design choice improves area utilisation by reducing memory to a 544-bit PASTA state. Another limiting factor in the design is the use of the data bus connecting the core to the peripherals. The latter, as a slave, handles the encryption start signal, the memory address for reading into RAM, reading the key and nonce, and writing the encrypted data, all on a single bus. This prevents simultaneous reading and writing of data blocks. So, the processing of one block must be completed before the next block can be started.

■ **Bitlength Comparison.** The performance stays the same for different bit lengths thanks to our design strategy. The area-time product increases as the area is more than doubled when the bit length is doubled. Therefore, we will consider the design with a bit-length of 17 bits for comparisons unless specified.

B. Performance Benchmarks.

In Sec. III, we described our design strategy tailored to offer high speed, which is limited by the Keccak permutation implementation. SHAKE128 instance of Keccak has a rate of 1,344 bits and generates 21 words (64-bit) after one permutation. Each permutation takes 24 clock cycles (cc). One round of PASTA-4 requires 32×4 pseudo-random state elements, one each for the two matrix multiplication and two round constant additions. PASTA-4 has five such rounds. Thus, a minimum of 31 Keccak permutation rounds is required.

A rejection sampling follows the pseudo-random number generation. In our case, we have a high rate of rejection sampling ($\approx 2\times$) for the stated prime- 65,537 (0×10001). Therefore, the sampling rate almost doubles, and we require, on average, 60 Keccak permutation rounds. Overall, the Keccak round function alone consumes 1,440 cc (60×24). Sampling 21 elements would add another 21 cc after every round. Therefore, the clock cycle almost doubles for a naive Keccak implementation. However, the Keccak implementation utilised in this work allows us to perform Keccak permutation in parallel with the squeeze operations, adding only an extra five clock cycles between two squeezes. This helps us reduce the clock cycles by half ($60 \cdot (21 + 5) = 1,560\text{cc}$).

The total clock cycle count is appended with $t = 32$ clock cycles required by the last remaining Mix operation after all the data has been generated, bringing the overall cycle count for PASTA-4 encryption to 1,592, as given in Tab. II. Similarly, for PASTA-3, the state size increases fourfold while operating for one less round. This gives the average number of Keccak calls as 186 and clock cycles for Keccak round function alone as 4,836cc ($186 \cdot (21 + 5)\text{cc}$). The entire PASTA-3 encryption costs 4,964cc due to an additional 128cc from the last mix operation.

TABLE III
COMPARISONS OF PASTA-4 PERFORMANCE, AREA WITH PRIOR WORKS.

Work	Platform	Area				Encr. [‡] (μ s)
		kLUT	kFF	DSP	BRAM	
[21]	Zynq US+	-	-	-	-	7.79k (1.91)
[22]	AlveoU250	1,179	1,036	12,288	828.5	16.9k (0.51)
[18]	Kintex-7	20.7	17.6	100	82.5	1.87k (0.46)
TW	Artix-7	23.7	11.1	64	0	21.2 (0.67)
[20]	12nm	0.06 mm^2				110k (26.9)
[19]	12nm [†]	0.11 mm^2				20k (4.88)
TW	7/28nm	0.03/0.24 mm^2				1.59 (0.05)
TW	65/130nm[†]	0.74/1.80 mm^2				15.9 (0.50)

[‡] Runtime is reported for one encryption and per element in bracket.

[†] refers to RISC-V SoC, and 'k' refers to 1,000.

These bounds on the number of clock cycles are based on the rejection sampling rate. Consequently, the number of clock cycles upon experimentation varies with a small deviation based on the initiating nonce and counter.

■ PASTA-3 vs. PASTA-4. Area-Performance Comparisons.

Tab. II lists the performance estimates upon experimentation on various platforms. While PASTA-3 consumes more runtime than PASTA-4 for their respective state size, PASTA-3 reports 22% less processing time than PASTA-4 for the same amount of data. In the previous section, we saw that PASTA-3 consumes approximately $3\times$ more area than PASTA-4. This shows us that PASTA-4 offers a better area-time trade-off than PASTA-3 and should be preferred for client-side devices irrespective of the choice of the implementation platform.

C. Comparisons with Prior Works

Tab. II reports that our design requires $857\text{--}3,439\times$ fewer clock cycles than CPU results presented in [9]. This translates to a speedup of $43\text{--}171\times$ as the CPU runs at $\approx 20\times$ higher clock frequency- 2.2GHz. Prior works [18]–[22] for FHE client-side encryption routine present designs targetting the PKE algorithm involving expensive polynomial arithmetic. No prior work has implemented new and more efficient *arithmetic* symmetric HHE enabling schemes. The parameters supported by the prior works are also insufficient to support bootstrapping in FHE. In contrast, implementing an HHE enabling SE scheme does not prevent bootstrapping.

① **FPGA works.** To ensure fair comparisons, the performance is also reported per element encrypted. The number of elements is 32 in our case, 2^{12} for [18], [21], and 2^{15} for [22]. Prior works [18], [21], [22] utilize Zynq UltraScale+, AlveoU250, and Kintex-7 FPGAs. These boards have $2\text{--}10\times$ more resources than our target Artix-7 board. Tab. III shows that our design consumes relatively less area and no BRAMs while delivering similar performance. Furthermore, we deliver similar performance while running our design at almost $2\text{--}3\times$ lower clock frequency, thus lowering the overall energy consumption. For ML inference applications encrypting low amounts of data (e.g., 32 coefficients), we deliver much better performance ($21.2\ \mu$ s) as FHE will necessitate the same amount of computations ($1,884\ \mu$ s) for any amount of data upto 2^{12} coefficients.

② **RISC-V works.** The existing RISC-V client-side FHE acceleration works [19], [20] present a hardware-software co-design targeting 1GHz clock frequency on 12nm Technology.

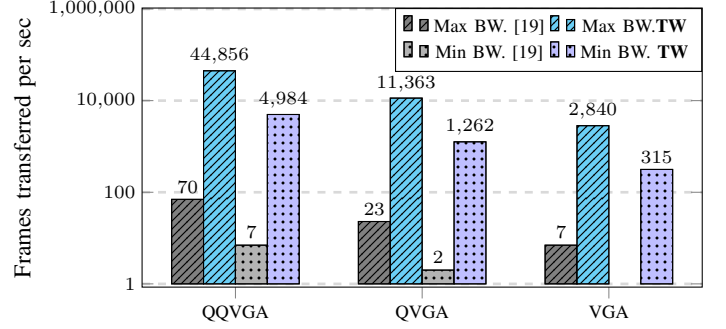


Fig. 8. Frames transferred per second for Maximum (112.5 MBps, left) and Minimum (12.5 MBps, right) 5G available bandwidth. The graphs are on a logarithmic scale. TW refers to this work.

Their parameters choice can pack 2^{12} elements per encryption. The comparisons in Tab. III reported with these works showcase that the design for HHE enabling SE schemes offers $98\text{--}338\times$ better performance as a standalone chip. The proposed RISC-V SoC on the old technology nodes (130nm, 65nm) performs $10\text{--}34\times$ better while targeting $10\times$ less clock frequency for low power. Note that the design consumes an area similar to [19] post-technology normalization.

V. APPLICATION BENCHMARK

We evaluate an application benchmark of the video frame encryption [19], [27] application, which sends videos to the cloud processors for surveillance [28]. The video is split into frames as per the required resolution, which can range from QVGA (160×120 pixels), QVGA (320×240 pixels), to VGA (640×480 pixels). A grayscale pixel requires 8 bits of storage. The available bandwidth for data transfer ranges between 12.5 and 112.5 MBps (e.g., mid-band 5G network [29]). Our results compared with [19] are reported in Fig. 8. RISE [19] can encode one QVGA frame per ciphertext ($N = 2^{14}, \log Q = 390$) and requires three such ciphertexts for a QVGA frame. One ciphertext size is 1.5MB ($2^{14} \cdot 2 \cdot 390$). Therefore, they can send 70 QVGA frames per second at the maximum 5G bandwidth. Our ciphertext for ($N = 2^5, \log q_0 = 33$) is only 132 Bytes in size ($2^5 \cdot 17$). Hence, we can send up to $712\times$ more frames per second. [19] cannot send a VGA frame at minimum bandwidth.

VI. CONCLUSION AND FUTURE SCOPE

In this work, we realized an HHE enabling SE scheme-PASTA in hardware. This is the first such realization of any efficient and new SE scheme defined over integer and presents results useful for promoting this field. It highlights certain problems, for example, the performance bottleneck due to heavy reliance on the XOF, and its significant area consumption. An interesting future scope is to implement the other HHE enabling SE schemes and show the impact of the changes across these schemes post-hardware realization. Another direction would be to analyze the effect of adding countermeasures against side-channel or fault analysis [30] and compare them against the cost of adding the same countermeasures on public-key encryption.

REFERENCES

- [1] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, USA, 2009.
- [2] W. Jung, E. Lee, S. Kim, J. Kim, N. Kim, K. Lee, C. Min, J. H. Cheon, and J. H. Ahn, "Accelerating fully homomorphic encryption through architecture-centric analysis and optimization," *IEEE Access*, vol. 9, pp. 98772–98789, 2021.
- [3] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings* (R. Safavi-Naini and R. Canetti, eds.), vol. 7417 of *Lecture Notes in Computer Science*, pp. 850–867, Springer, 2012.
- [4] C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger, "Rasta: A cipher with low anddepth and few ands per bit," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I* (H. Shacham and A. Boldyreva, eds.), vol. 10991 of *Lecture Notes in Computer Science*, pp. 662–692, Springer, 2018.
- [5] P. Méaux, A. Journault, F. Standaert, and C. Carlet, "Towards stream ciphers for efficient FHE with low-noise ciphertexts," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I* (M. Fischlin and J. Coron, eds.), vol. 9665 of *Lecture Notes in Computer Science*, pp. 311–343, Springer, 2016.
- [6] P. Méaux, C. Carlet, A. Journault, and F. Standaert, "Improved filter permutations for efficient FHE: better instances and implementations," in *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings* (F. Hao, S. Ruj, and S. S. Gupta, eds.), vol. 11898 of *Lecture Notes in Computer Science*, pp. 68–91, Springer, 2019.
- [7] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey, "Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression," *J. Cryptol.*, vol. 31, no. 3, pp. 885–916, 2018.
- [8] J. Ha, S. Kim, W. Choi, J. Lee, D. Moon, H. Yoon, and J. Cho, "Masta: An he-friendly cipher using modular arithmetic," *IEEE Access*, vol. 8, pp. 194741–194751, 2020.
- [9] C. Dobraunig, L. Grassi, L. Helming, C. Rechberger, M. Schafnegg, and R. Walch, "Pasta: A case for hybrid homomorphic encryption," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 3, pp. 30–73, 2023.
- [10] J. Cho, J. Ha, S. Kim, B. Lee, J. Lee, J. Lee, D. Moon, and H. Yoon, "Transciphering framework for approximate homomorphic encryption," in *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III* (M. Tibouchi and H. Wang, eds.), vol. 13092 of *Lecture Notes in Computer Science*, pp. 640–669, Springer, 2021.
- [11] J. Ha, S. Kim, B. Lee, J. Lee, and M. Son, "Rubato: Noisy ciphers for approximate homomorphic encryption," in *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I* (O. Dunkelman and S. Dziembowski, eds.), vol. 13275 of *Lecture Notes in Computer Science*, pp. 581–610, Springer, 2022.
- [12] NIST, "Fips pub 197: Advanced encryption standard (aes)," p. 311, 2001.
- [13] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schl  ffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," *J. Cryptol.*, vol. 34, no. 3, p. 33, 2021.
- [14] Aikata, A. C. Mert, M. Imran, S. Pagliarini, and S. S. Roy, "Kali: A crystal for post-quantum security using kyber and dilithium," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 70, no. 2, pp. 747–758, 2023.
- [15] A. Aikata, A. C. Mert, S. Kwon, M. Deryabin, and S. S. Roy, "Reed: Chiplet-based scalable hardware accelerator for fully homomorphic encryption," 2023.
- [16] A. C. Mert, Aikata, S. Kwon, Y. Shin, D. Yoo, Y. Lee, and S. S. Roy, "Medha: Microcoded Hardware Accelerator for computing on Encrypted data," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 1, pp. 463–500, 2023.
- [17] R. Geelen, M. Van Beirendonck, H. V. L. Pereira, B. Huffman, T. McAuley, B. Selfridge, D. Wagner, G. Dimou, I. Verbauwhe, F. Vercauteren, and D. W. Archer, "BASALISC: Flexible Asynchronous Hardware Accelerator for Fully Homomorphic Encryption," 2022.
- [18] F. Krieger, F. Hirner, A. C. Mert, and S. S. Roy, "Aloha-he: A low-area hardware accelerator for client-side operations in homomorphic encryption," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2024.
- [19] Z. Azad, G. Yang, R. Agrawal, D. Petrisko, M. B. Taylor, and A. Joshi, "RISE: RISC-V soc for en/decryption acceleration on the edge for homomorphic encryption," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 31, no. 10, pp. 1523–1536, 2023.
- [20] Z. Azad, G. Yang, R. Agrawal, D. Petrisko, M. B. Taylor, and A. Joshi, "RACE: RISC-V soc for en/decryption acceleration on the edge for homomorphic computation," in *ISLPED '22: ACM/IEEE International Symposium on Low Power Electronics and Design, Boston, MA, USA, August 1 - 3, 2022*, ACM, 2022.
- [21] S. D. Matteo, M. L. Gerfo, and S. Saponara, "VLSI design and FPGA implementation of an NTT hardware accelerator for homomorphic seal-embedded library," *IEEE Access*, vol. 11, pp. 72498–72508, 2023.
- [22] J. Lee, P. Duong-Ngoc, and H. Lee, "Configurable encryption and decryption architectures for ckks-based homomorphic encryption," *Sensors*, vol. 23, no. 17, p. 7389, 2023.
- [23] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings* (P. Rogaway, ed.), vol. 6841 of *Lecture Notes in Computer Science*, pp. 222–239, Springer, 2011.
- [24] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, "The LED block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings* (B. Preneel and T. Takagi, eds.), vol. 6917 of *Lecture Notes in Computer Science*, pp. 326–341, Springer, 2011.
- [25] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.
- [26] "Ibex risc-v core."
- [27] M. A. Usman, M. R. Usman, and S. Y. Shin, "An intrusion oriented heuristic for efficient resource management in end-to-end wireless video surveillance systems," in *15th IEEE Annual Consumer Communications & Networking Conference, CCNC 2018, Las Vegas, NV, USA, January 12-15, 2018*, pp. 1–6, IEEE, 2018.
- [28] B. Lagesse, G. Nguyen, U. Goswami, and K. Wu, "You had to be there: Private video sharing for mobile phones using fully homomorphic encryption," in *19th IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2021, Kassel, Germany, March 22-26, 2021*, pp. 730–735, IEEE, 2021.
- [29] S. R. Thummalur, M. Ameen, and R. K. Chaudhary, "Four-port mimo cognitive radio system for midband 5g applications," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 8, pp. 5634–5645, 2019.
- [30] Aikata, D. Saha, and S. S. Roy, "SASTA: ambushing hybrid homomorphic encryption schemes with a single fault," *IACR Cryptol. ePrint Arch.*, p. 41, 2024.