YuX: Finite Field Multiplication Based Block Ciphers for Efficient FHE Evaluation

Fen Liu, Yongqiang Li, Huiqin Chen, Lin Jiao, Ming Luo, and Mingsheng Wang

***Abstract*— With the growing practical applications of fully homomorphic encryption (FHE), secure multi-party computation (MPC), and zero-knowledge proofs (ZK), there has been an increasing need to design and analyze symmetric primitives that have low multiplication complexity and depth. In this paper, we propose a permutation constructed upon a 4-round nonlinear feedback resistor over $\mathbb{F}_q^4$. Our proposed permutation has a multiplication depth of 2 and a multiplication complexity of 4. Significantly, its maximum differential/linear probability is bounded by q^{-2}. Based on this nonlinear function, we propose a new family of block ciphers over $\mathbb{F}_q^{16}$ called YuX, whose decryption circuit is highly efficient for FHE evaluation. We further provide specific instantiations, denoted as $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$, wherein q takes the form of either 2^n or a prime p, respectively. Furthermore, we conduct a comprehensive security analysis of YuX within certain parameters against various cryptanalysis methods employing automatic analysis tools, including the differential attack, linear attack, impossible differential attack, zero-correlation attack, and integral attack, as well as Gröbner basis and linearization attacks. Our research indicates that YuX maintains a robust security margin against those attacks. Finally, we present a detailed implementation of $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$ employing the BGV homomorphic encryption scheme. In comparison to ciphers over a field of characteristic 2, the outcomes evince that $\mathsf{Yu_2X}$-8 (over $\mathbb{F}_{2^8}^{16}$) and $\mathsf{Yu_2X}$-16 (over $\mathbb{F}_{2^{16}}^{16}$) achieve remarkably competitive throughputs, boasting performance approximately 12 times, 17 times, and 9 times superior to AES-128, CHAGHRI, and LowMC-128 (under 128-bit security), respectively. Furthermore, when juxtaposed with ciphers over a field of characteristic p, the outcomes affirm that the throughput of $\mathsf{Yu_pX}$-65537 (over $\mathbb{F}_{65537}^{16}$) retains considerable competitiveness, registering an approximate fivefold enhancement relative to HERA. Evidently, YuX exhibits superior throughput compared to a majority of symmetric ciphers within this category.**

***Index Terms*— Permutation, block cipher, FHE, low multiplication complexity, low multiplication depth.**

I. Introduction

With the rapid development of technologies, data management, storage, transformation, and dissemination methods have significantly changed. Data outsourcing has become increasingly popular, and it is essential to strictly guarantee the confidentiality and integrity of outsourced data. Some solutions to this problem are to employ privacy-preserving cryptographic primitives, such as fully homomorphic encryption (FHE), secure multi-party computation (MPC), and zero-knowledge proofs (ZK).

FHE is a cryptographic primitive that enables computation over encrypted data directly and obtaining the same encrypted results as if performing on the plaintext. Therefore, using FHE to encrypt the data before sending it to the server can protect data privacy and allow the server to perform necessary computations. The most common FHE schemes are divided into three categories: the first group is primarily used in Boolean circuits (i.e., encrypt boolean values in $\mathbb{Z}_2$), including the FHEW [1] and TFHE schemes [2]; the second group supports operations over finite fields (i.e., encrypt plaintext in $\mathbb{Z}_p$) including BGV [3] and BFV [4], [5] schemes; and then the third and most recent group is represented by the CKKS scheme [6], which provides approximate arithmetic over real and complex numbers. All these FHE schemes are based on the Ring Learning With Errors (Ring-LWE) hardness assumption.

However, one major challenge with FHE schemes is the large ciphertext size, which can result in high communication overhead in cases of bulk data transfer. To address this challenge, the Hybrid Homomorphic Encryption (HHE) framework was proposed in [7], which is also called a transciphering framework [8]. The transciphering framework allows the client to use a symmetric cipher to encrypt the private data instead of homomorphic encryption, which can significantly reduce communication overhead between the client and server. The server then homomorphically performs a symmetric decryption circuit to transform the symmetric ciphertext into homomorphic ciphertext using the homomorphically encrypted key received from the client.

Most traditional symmetric ciphers like AES [9] and PRINCE [10] are not well-suited for homomorphic evaluation as their arithmetic complexity, particularly multiplication depth, is not explicitly designed for such applications. In these applications, linear components can be considered "free", while nonlinear components can cause rapid noise growth and significantly increase execution time. Additionally, the noise grows exponentially with the multiplication depth of

Manuscript received 9 March 2023; revised 10 August 2023; accepted 19 December 2023. Date of publication 3 January 2024; date of current version 23 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 12371525. *(Corresponding author: Yongqiang Li.)*

Fen Liu, Yongqiang Li, Huiqin Chen, Ming Luo, and Mingsheng Wang are with the Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100045, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: liufenxd@gmail.com; yongq.lee@gmail.com).

Lin Jiao is with the State Key Laboratory of Cryptology, Beijing 100878, China.

Communicated by T. Johansson, Associate Editor for Sequences and Cryptography.

Digital Object Identifier 10.1109/TIT.2024.3349414

the circuit. Thus, nonlinear components are the most significant performance bottleneck in these applications. Our work aims to address this challenge by proposing a new construction of nonlinear functions with lower multiplication complexity and depth, while still maintaining good cryptography properties. We then use this construction to design a block cipher that improves the efficiency of homomorphic evaluations.

A. Low Multiplication Complexity and Depth Ciphers

The multiplicative complexity refers to the count of multiplications (AND gates) present in a circuit, while the multiplicative depth corresponds to the maximum number of consecutive multiplication operations in the circuit. Some new symmetric schemes have been especially proposed for FHE schemes, MPC protocols, and ZK protocols. The main design criterion of these ciphers is to reduce the multiplication complexity and multiplication depth as much as possible.

1) Ciphers for $\mathbb{F}_2$: Some of these ciphers are designed based on operations over $\mathbb{F}_2$, such as LowMC [11], Kreyvium [12], FLIP [13], Rasta [14], Dasta [15], Fasta [16], etc. These ciphers are designed to minimize the number of AND gates and the AND depth, and the nonlinear functions of these ciphers are usually quadratic S-boxes or quadratic Boolean functions. Meanwhile, insights on cryptanalysis of the new ciphers are constantly proposed to analyze the security of those ciphers.

LowMC is a family of block ciphers with a 3-bit S-box and randomly generated round linear mappings. The S-box of LowMC contains 3 AND gates and the AND depth is 1. The first version of LowMC was attacked by an optimized variant of the interpolation attack [17], and the 80-bit key weak instances were fully broken in about 2^{57} times, using 2^{39} chosen plaintexts. Then based on the higher-order differential cryptanalysis, full key recovery of the 80-bit key instances for 9 rounds (fully 11 rounds) had been got with a data complexity of 2^{59} [18], which resulted in LowMCv2 with an increased number of rounds. In [19], fully-round versions of LowMCv2 with few S-boxes per round instances were broken by adopting the enumeration of differential attacks. Afterward, the designers proposed an updated round formula for LowMC, which is called LowMCv3.

Kreyvium is a family of NFSR-based stream ciphers. The nonlinear feedback function of Kreyvium is quadratic Boolean functions with AND depth 1. Several papers have studied the analysis of Kreyvium. In [20], a Zero-Sum distinguisher of Kreyvium on 872 rounds has been found with a cube of size 61, and later the results are improved to 873 rounds using division property [21]. In [22], the key recovery attacks on 891 rounds were reported. It should be noted that these attacks do not threaten the security of Kreyvium.

FLIP is a family of stream ciphers based on the filter generator without the register update part. The filter function is designed with quadratic Boolean functions and triangular Boolean functions with AND depth larger than 1. In the case of the early version of FLIP, based on the weakness of structure and guess-and-determine attacks, the full key recovery can be performed in 2^{54} basic operations [23].

Rasta is a family of stream ciphers. Its nonlinear layer is a n-bit quadratic permutation and the AND depth is 1, where n is odd. Its affine layers are randomly generated for encrypting each block. Dasta is a variant of Rasta substituting the linear layer with a permutation followed by a deterministic matrix multiplication which is faster than Rasta on the offline side. Fasta also can be considered as a variant of Rasta, whose parameters are chosen considering the parallelism offered by BGV to optimize the homomorphic implementation with HElib. Combining the structural weaknesses of Rasta and Dasta, the security margins of some recommended parameters are reduced to only 1 round based on improved algebraic cryptanalysis [24].

2) Ciphers for $\mathbb{F}_q$: There are other ciphers based on operations over large fields $\mathbb{F}_q$ (q is a prime number p or a power of 2) since many protocols naturally support operations in a larger field $\mathbb{F}_q$, and converting operations over $\mathbb{F}_q$ into bit operations is expensive. Examples include MiMC [25], GMiMC [26], Jarvis [27], CHAGHRI [28], Masta [29], Pasta [30], etc. Those ciphers working over large fields can easily resist differential cryptanalysis. However, there are some new cryptanalysis techniques for these ciphers, and the algebraic attacks usually have effective results.

MiMC is a block cipher with nonlinear function x^3 for $x \in \mathbb{F}_q$. It has several different versions, including the version of the Feistel structure and the version of the direct iteration of the round function over $\mathbb{F}_q$, where q is either a prime p or a power of 2. Moreover, a generalized version of MiMC named GMiMC was later proposed to provide efficient performance for PQ-secure signature schemes [26]. Based on a generalization of higher-order differential cryptanalysis, key recovery attacks on the full-round version of MiMC over $\mathbb{F}_{2^n}$ were presented in [31].

CHAGHRI is an SPN-based block cipher designed over $\mathbb{F}_{2^{63}}^3$ and the nonlinear function is $x^{2^{35}+1}$ for $x \in \mathbb{F}_{2^{63}}$. With higher-order differential cryptanalysis, a full 8-round attack can be achieved with time and data complexity of 2^{28}. Meanwhile, a new technique named coefficient grouping has been proposed to evaluate CHAGHRI, and a 13-round distinguisher with time and data complexity of 2^{63} has been found [32]. Hence, the round number of CHAGHRI can not offer adequate security. In the new version of CHAGHRI, the affine permutation after the S-box is adjusted to resist this attack [28].

In the case of stream ciphers Masta and Pasta, both are new variants of Rasta that use modular arithmetic to support HE schemes over $\mathbb{F}_p$ (a non-binary plaintext space like Rasta), where p is a large prime number. There have been no released cryptanalysis results for Masta and Pasta up to now. There are still some new stream ciphers that can be used to combine the CKKS and BFV homomorphic encryption schemes to encrypt real numbers. Such as stream cipher HERA [8] and noisy ciphers Rubato [33]. However, with algebraic attacks typically defined over fields, an attack on full Rubato has been found [34].

B. Our Contributions

In this paper, we present a permutation over $\mathbb{F}_q^4$ with lower multiplication complexity and depth. The permutation is constructed by iterating a nonlinear feedback register in 4 steps, with each step containing one multiplication over $\mathbb{F}_q$. The multiplication complexity and depth of the permutation are 4 and 2, respectively. We prove that the maximum differential/linear probability of the permutation over $\mathbb{F}_q^4$ is less than or equal to q^{-2}.

We then utilize the proposed permutation over $\mathbb{F}_q^4$ to design a family of block ciphers called YuX, which has a highly efficient decryption circuit for FHE evaluation. It is based on an SPN structure, and each round transformation contains four parallel S-boxes that are the compositional inverse of the proposed permutation over $\mathbb{F}_q^4$, followed by an MDS mapping over $(\mathbb{F}_q^4)^4$. As illustrations, we present specific instantiations of YuX over finite fields $(\mathbb{F}_{2^n}^4)^4$ and prime fields $(\mathbb{F}_p^4)^4$, denoted as $\mathsf{Yu}_2\mathsf{X}$ and $\mathsf{Yu}_\mathsf{p}\mathsf{X}$ correspondingly.

Furthermore, we conduct the security analysis of some YuX specifications using SAT and MILP tools. In the context of 128-bit security, the longest distinguishers we found for the differential attack, linear attack, impossible differential attack, zero-correlation attack, and integral attack are 3, 3, 2, 2, and 6 rounds, respectively. Additionally, we present the cryptanalysis results of YuX against algebraic attacks, including Gröbner basis attack and linearization attack. The results demonstrate that YuX has a sufficient secure margin.

Finally, We utilize the BGV scheme implemented in the open-source homomorphic encryption library HElib to implement $\mathsf{Yu}_2\mathsf{X}$ and $\mathsf{Yu}_\mathsf{p}\mathsf{X}$. Our implementation employs a state-sliced approach, enabling us to represent multiple blocks in parallel using multiple ciphertexts. This reduces the number of required multiplications and automorphism operations for each block on average, leading to an improvement in the overall efficiency of the homomorphic evaluation. In terms of throughput for FHE evaluation, $\mathsf{Yu}_2\mathsf{X}$-8 and $\mathsf{Yu}_2\mathsf{X}$-16 achieve rates of 12.80 KB/min and 12.33 KB/min, which are approximately 12, 17 and 9 times higher than that of AES-128, CHAGHRI and LowMC-128 (with 128-bit security), respectively. Furthermore, the throughput of $\mathsf{Yu}_\mathsf{p}\mathsf{X}$ achieves 858 KB/min. Notably, the throughput of YuX exceeds that of a considerable portion of stream ciphers.

C. Organization

In Sect. II, we review the BGV-based homomorphic encryption scheme used in later sections. We give the construction of a permutation over $\mathbb{F}_q^4$ and characterize its differential uniform and linearity in Sect. III. We give the detail of YuX in Sect. IV, followed by the design rationale in Sect. V. Then, we conduct the security analysis of YuX against various attacks in Sect. VI. We present the specification of implementation in Sect. VII. Finally, a short conclusion is given in Sect. VIII.

II. Preliminaries

A. Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) schemes allow arbitrary functions over encrypted data without the decryption key. Nowadays, FHE schemes are based on the hardness of the LWE problem and are optimized by using Ring-LWE. In this paper, we consider the BGV scheme [3], and the implementation is done by employing the HElib library.[1]

1) BGV: The Brakerski-Gentry-Vaikuntanathan (BGV) scheme is one of the primary homomorphic encryption (HE) schemes to compute over encrypted data based on the hardness of the Ring-LWE problem. Let m be a positive integer. Let quotient ring $\mathbb{A}$ be $\mathbb{Z}[x]/(\Phi_m(x))$, where $\Phi_m(x)$ is the m-th cyclotomic polynomial, $\mathbb{Z}$ is the set of integers. Let p be a prime number, and the plaintext works on the ring $\mathbb{A}_p = \mathbb{A}/(p \cdot \mathbb{A})$ while the initial ciphertext works on the ring $\mathbb{A}_q = \mathbb{A}/(q \cdot \mathbb{A})$. For a plaintext $\mathsf{m} \in \mathbb{A}_p$, the ciphertext associated with secret s is a pair of polynomials $\boldsymbol{c} = (c_0, c_1) \in \mathbb{A}_q^2$, which also can be represented as

$$\boldsymbol{c} = (as + p \cdot e + \mathsf{m}, a) \mod q,$$

where $a \leftarrow \mathcal{U}(\mathbb{A}_q)$, $s \leftarrow \mathcal{HWT}(w)$, $e \leftarrow \mathcal{DG}(\sigma^2)$, $\mathcal{U}(\mathbb{A}_q)$ is the uniform distribution over $\mathbb{A}_q$, $\mathcal{HWT}(w)$ is the distribution sampling a ternary secret with Hamming weight w, $\mathcal{DG}(\sigma^2)$ is the distribution that samples an element in $\mathbb{A}$ whose each coefficient is independently from the discrete Gaussian distribution with standard deviation σ, and left arrow "$\leftarrow$" is denoted as sampling a random element from the corresponding distribution.

2) Packing: The BGV scheme supports some basic operations, including homomorphic addition, scalar multiplication (pt-ct multiplication operation), multiplication, and automorphism. The BGV scheme also supports SIMD operations [35], which encode a number of independent small finite fields into one plaintext polynomial. The homomorphic operations over plaintext polynomials can always influence these independent finite fields. Let $p > 1$ be a prime number, and we assume that the monic cyclotomic polynomial $\Phi_m(x)$ has degree $N = \phi(m)$, where ϕ is Euler's totient function. Then $\Phi_m(x)$ can be split into r distinct irreducible polynomials with degree of $d = N/r$, such as $\Phi_m(x) = \prod_{i=1}^{r} F_i(x) \pmod p$. The ring $\mathbb{A}_p$ (i.e., plaintext space. p is a prime, so $\mathbb{A}_p$ is a field too.) can be decomposed to some smaller fields $\{\mathbb{A}_p/F_i(x)\}_{i=1}^r$ using the Chinese Remainder Theorem (CRT), where each one of them is isomorphic to a finite field $\mathbb{F}_{p^d}$. Then r elements of $\mathbb{F}_{p^d}$ or its subfield $\mathbb{F}_{p^{d'}}$ can be computed parallelly, where r is the number of slots, and d' is one of the d's positive factors. The homomorphic addition and multiplication operations over these slots are obviously supported after this packing.

Another important operation in our paper is the automorphism, which homomorphically maps a polynomial $a(x) \in \mathbb{A}_p$ to $a^{(i)}(x) = a(x^i) \pmod{\Phi_m(x)}$, where $i \in \left(\mathbb{Z}/(m \cdot \mathbb{Z})\right)^*$. By careful setting [36], the automorphism can act two meaningful operations on slots. If i is not a power of p, the automorphism transformation acts a cyclic rotation over these slots, i.e., $(\alpha_0, \alpha_1, \cdots, \alpha_{r-1}) \mapsto (\alpha_j, \alpha_{j+1}, \cdots, \alpha_{j-1}) \in (\mathbb{F}_{p^d})^r$, $j \in \mathbb{Z}/(r \cdot \mathbb{Z})$. If i is a power of p, the automorphism transformation acts an automorphism over each slot, i.e., $(\alpha_0, \alpha_1, \cdots, \alpha_{r-1}) \mapsto (\alpha_0^{p^j}, \alpha_1^{p^j}, \cdots, \alpha_{r-1}^{p^j}) \in (\mathbb{F}_{p^d})^r$, $j \in \mathbb{Z}/(d \cdot \mathbb{Z})$, which is also called Frobenius automorphism.

[1] https://github.com/homenc/HElib

TABLE I
THE STATISTICAL CRYPTANALYSIS OF SPECIFIC YUX SPECIFICATIONS

Ciphers	differential attack	linear attack	impossible differential attack	zero-correlation attack	integral attack
Yu_2X-8 and Yu_2X-16	3	3	2	2	6
Yu_pX-65537	2	2	2	2	6

B. Cryptography Properties of an S-Box

Throughout this paper, $\mathbb{F}_q$ denotes the finite field with q elements, and its character p equals 2 or an odd prime. For a polynomial $F(x) \in \mathbb{F}_q[x]$, its differential uniformity $\Delta(F)$ is defined as

$$\Delta(F) = \max_{a,b\in\mathbb{F}_q, a\neq 0} |\{x \in \mathbb{F}_q \mid F(x) - F(x-a) = b\}|,$$

and $F(x)$ is called differential δ-uniform when $\Delta(F) = \delta$ [37]. The Walsh transform of F is defined as

$$\lambda_F(u, v) = \sum_{x\in\mathbb{F}_q} \chi(vF(x) - ux),$$

where $u, v \in \mathbb{F}_q$, $\chi(x) = e^{\frac{2\pi i}{p}\mathrm{Tr}_1^m(x)}$, p is the character of $\mathbb{F}_q$ and m is the integer such that $p^m = q$. The linearity of F is the highest 2-norm of its Walsh coefficients:

$$\mathcal{L}(F) = \max_{v\in\mathbb{F}_q^*, u_q\in\mathbb{F}} |\lambda_F(u, v)|.$$

The maximum differential probability and linear probability (squared correlation) is defined as

$$DP^F = \frac{\Delta(F)}{q}, LP^F = \left(\frac{\mathcal{L}(F)}{q}\right)^2$$

respectively [38], [39].

III. A NONLINEAR FUNCTION OVER $\mathbb{F}_q^4$ AND ITS CRYPTOGRAPHY PROPERTIES

In this section, we give a construction of a permutation over $\mathbb{F}_q^4$ with low multiplication depth, and we also provide a comprehensive analysis of the differential uniformity and linearity of this permutation. Note that when the character of $\mathbb{F}_q$ is 2, then both the operations addition "+" and minus "−" are the operation "⊕" and $2\beta = 0$ for any $\beta \in \mathbb{F}_q$.

A. The Construction of a Permutation Over $\mathbb{F}_q^4$ With Low Multiplication Depth

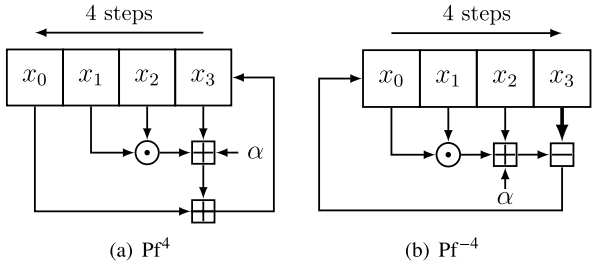
Let $x_i \in \mathbb{F}_q, 0 \le i \le 3$, f be a vectorial function from $\mathbb{F}_q^3$ to $\mathbb{F}_q$. It is easy to see that the following mapping

$$\mathrm{Pf}(x_0, x_1, x_2, x_3) = (x_1, x_2, x_3, x_0 + f(x_1, x_2, x_3))$$

is a permutation over $\mathbb{F}_q^4$. Based on the multiplication over finite fields, we propose the following construction.

Construction 1: Let $x_i \in \mathbb{F}_q, 0 \le i \le 3$, $\alpha \in \mathbb{F}_q$. Let $\mathrm{Pf} : \mathbb{F}_q^4 \mapsto \mathbb{F}_q^4$ as follows

$$\mathrm{Pf}(x_0, x_1, x_2, x_3) = (x_1, x_2, x_3, x_0 + x_1x_2 + x_3 + \alpha),$$



(a) Pf^4 (b) Pf^{-4}

Fig. 1. The permutation over $\mathbb{F}_q^4$ and its compositional inverse.

where x_1x_2 means x_1 multiply x_2 over $\mathbb{F}_q$. Then

$$\mathrm{Pf}^4(x_0, x_1, x_2, x_3)$$

is a permutation over $\mathbb{F}_q^4$, where $\mathrm{Pf}^i = \mathrm{Pf} \circ \mathrm{Pf}^{i-1}$ for $i \ge 2$.

The implementation cost of $\mathrm{Pf}(x_0, x_1, x_2, x_3)$ involves a multiplication over the finite field $\mathbb{F}_q$, two additions over $\mathbb{F}_q$ and an addition with a constant. Given the equality

$$\mathrm{Pf}^2(x_0, x_1, x_2, x_3) = (x_2, x_3, x_0 + x_1x_2 + x_3 + \alpha, x_1 + x_2x_3 + x_0 + x_1x_2 + x_3 + 2\alpha),$$

it requires two multiplications to directly compute Pf^2. However, the two multiplications can be performed simultaneously, which means that the multiplication depth of Pf^2 is only 1 and the multiplication complexity is 2. Additionally, it is worth noting that $\mathrm{Pf}^4 = (\mathrm{Pf}^2)^2$, which allows us to iterate the circuit of Pf^2 twice to compute Pf^4. Consequently, the multiplication depth and complexity of Pf^4 are 2 and 4, respectively.

As for the compositional inverse of Pf^4, note that

$$\mathrm{Pf}^{-1}(x_0, x_1, x_2, x_3) = (x_3 - x_0x_1 - x_2 - \alpha, x_0, x_1, x_2),$$

then

$$(\mathrm{Pf}^4)^{-1}(x_0, x_1, x_2, x_3) = (\mathrm{Pf}^{-1})^4(x_0, x_1, x_2, x_3).$$

B. The Cryptography Properties of Pf^4

The Pf^4 and $(\mathrm{Pf}^4)^{-1}$ are permutations over $\mathbb{F}_q^4$, and they are the compositional inverse of each other. Then they are CCZ-equivalent and have the same differential uniformity and linearity [40]. In the following, we only characterize these properties of Pf^4.

Theorem 1: Let Pf^4 be the permutation over $\mathbb{F}_q^4$ in Construction 1. Then the differential uniformity of Pf^4 is less than or equal to q^2.

Proof: Denote $\bar{x} = (x_0, x_1, x_2, x_3)$, and $\bar{0} = (0, 0, 0, 0)$. Let $\bar{a} = (a_0, a_1, a_2, a_3), \bar{b} = (b_0, b_1, b_2, b_3) \in \mathbb{F}_q^4$. We need to prove that for any $\bar{0} \neq \bar{a}, \bar{b} \in \mathbb{F}_q^4$, the difference equation

$\mathrm{Pf}^4(\bar{x}) - \mathrm{Pf}^4(\bar{x}-\bar{a}) = \bar{b}$ has at most q^2 solutions in $\mathbb{F}_q^4$. Note that the above differential equation holds if and only if

$$\mathrm{Pf}^{-2}(\mathrm{Pf}^4(\bar{x}-\bar{a})) = \mathrm{Pf}^{-2}(\mathrm{Pf}^4(\bar{x}) - \bar{b}).$$

Let $\bar{y} = (y_0, y_1, y_2, y_3) = \mathrm{Pf}^4(\bar{x})$. Since $\mathrm{Pf}^2(\bar{x}) = \mathrm{Pf}^{-2}(\bar{y})$, then the above equation is equivalent to

$$\mathrm{Pf}^2(\bar{x}) - \mathrm{Pf}^2(\bar{x}-\bar{a}) = \mathrm{Pf}^{-2}(\bar{y}) - \mathrm{Pf}^{-2}(\bar{y}-\bar{b}).$$

Note that

$$\begin{aligned}\mathrm{Pf}^2(x_0,x_1,x_2,x_3)=&(x_2, x_3, x_0 + x_1x_2 + x_3 + \alpha,\\ &x_1 + x_2x_3 + x_0 + x_1x_2 + x_3 + 2\alpha),\\ \mathrm{Pf}^{-2}(y_0,y_1,y_2,y_3)=&(y_2 - (y_3 - y_0y_1 - y_2 - \alpha)y_0 - y_1\\ &-\alpha, y_3 - y_0y_1 - y_2 - \alpha, y_0, y_1),\end{aligned}$$

then the difference equation becomes

$$\begin{cases} a_2 = -b_0y_3 - b_3y_0 + b_1y_0^2 + 2b_0y_0y_1 - 2b_0b_1y_0\\ \quad -b_0^2y_1 + b_0y_2 + b_2y_0 + \beta\\ a_3 = -b_0y_1 - b_1y_0 + b_0b_1 - b_2 + b_3\\ b_0 = a_1x_2 + a_2x_1 - a_1a_2 + a_0 + a_3\\ b_1 = a_2x_3 + a_3x_2 + a_1x_2 + a_2x_1 - a_2a_3 - a_1a_2\\ \quad +a_0 + a_1 + a_3,\end{cases}$$

where $\beta = b_0b_3 + b_0^2b_1 - b_0b_2 + b_2 - b_1 + \alpha b_0$. The fourth equation minus the third equation, we get

$$\begin{cases} a_2 = -b_0y_3 - b_3y_0 + b_1y_0^2 + 2b_0y_0y_1 - 2b_0b_1y_0\\ \quad -b_0^2y_1 + b_0y_2 + b_2y_0 + \beta & (1a)\\ a_3 = -b_0y_1 - b_1y_0 + b_0b_1 - b_2 + b_3 & (1b)\\ b_0 = a_1x_2 + a_2x_1 - a_1a_2 + a_0 + a_3 & (1c)\\ b_1 - b_0 = a_2x_3 + a_3x_2 - a_2a_3 + a_1. & (1d)\end{cases}$$

Then we have the following cases.

Case 1: $a_2 \neq 0$. From Equ.(1c) and Equ.(1d), we have $x_1 = -a_2^{-1}(a_1x_2 - a_1a_2 + a_0 + a_3 - b_0)$ and $x_3 = -a_2^{-1}(a_3x_2 - a_2a_3 + a_1 + b_0 - b_1)$.

This means x_1 and x_3 are determined uniquely by x_2. Note that the free variables x_0 and x_2 are chosen from the finite field $\mathbb{F}_q$, then the number of solutions of the above system of equations is less than or equal to q^2.

Case 2: $b_0 \neq 0$. Similarly, from Equ.(1b), we have $y_1 = b_0^{-1}(-b_1y_0 + b_0b_1 - b_2 + b_3 - a_3)$, which means y_1 is determined uniquely by y_0. From Equ.(1a), we have $y_3 = b_0^{-1}(-b_3y_0 + b_1y_0^2 + 2b_0y_0y_1 - 2b_0b_1y_0 - b_0^2y_1 + b_0y_2 + b_2y_0 + \beta - a_2)$, which means y_3 can be determined by y_2 uniquely when y_0, y_1 are assigned some values. Thus we can choose $y_0 \in \mathbb{F}_q$ and get y_1 from the above equality, then we chose $y_2 \in \mathbb{F}_q$ freely and get y_3 from the value of y_0, y_1, y_2. This means y_0, y_2 are freely in $\mathbb{F}_q$ and hence the number of solutions of the above system of equations is less than or equal to q^2.

Case 3: $a_2 = 0, b_0 = 0$. Then the above system of equations becomes

$$\begin{cases} b_3y_0 - b_1y_0^2 - b_2y_0 = -b_1 + b_2\\ \qquad\qquad b_1y_0 = -b_2 + b_3 - a_3\\ \qquad\qquad a_1x_2 = -a_0 - a_3\\ \qquad\qquad a_3x_2 = b_1 - a_1\end{cases}$$

We claim that $a_1 = 0$ and $a_3 = 0$ can not hold simultaneously. Otherwise, we can get $a_3 = a_0 = 0$ from the third equation. This contradicts with the condition that $\bar{a} = (a_0, a_1, a_2, a_3) \neq \bar{0}$. Then we have the following two cases:

Case 3.1: $a_1 \neq 0$. Then we have $x_2 = -a_1^{-1}(a_0 + a_3)$, which means x_2 is a constant in $\mathbb{F}_q$ that determined by $\bar{a}$. We claim that $b_1 = 0$ and $a_3 = 0$ can not hold simultaneously. Otherwise, we have $a_1 = b_1 = 0, b_2 = b_3, b_2 = 0$. This implies $\bar{b} = (0,0,0,0)$, which contradicts with the fact that Pf^4 is a permutation over $\mathbb{F}_q^4$.

When $b_1 \neq 0$, then we can get $y_0 = b_1^{-1}(-b_2 + b_3 - a_3)$ from the second equation. When $b_1 = 0$ and $a_3 \neq 0$, then we can get that $b_2 + b_3 = a_3 \neq 0$ from the second equation. Substitute it to the first equations, we can get that $y_0 = a_3^{-1}(-b_1 + b_2)$. This means y_0 is also a constant in $\mathbb{F}_q$ that determined by $\bar{a}, \bar{b}$. Note that $y_0 = x_0 + x_1x_2 + x_3 + \alpha$, then there are at most two variables in x_0, x_1, x_3 can chosen from $\mathbb{F}_q$ freely. Therefore, the number of solutions of the above system of equations is less than or equal to q^2.

Case 3.2: $a_1 = 0$. We claim that $a_3 \neq 0$, otherwise, we have $a_0 = 0, b_1 = 0, b_2 = b_3, b_2 = 0$. This implies $\bar{b} = (0,0,0,0)$, which contradicts with the fact that Pf^4 is a permutation over $\mathbb{F}_q^4$. When $b_1 \neq 0$, then $y_0 = b_1^{-1}(-b_2 + b_3 - a_3)$, which is a constant in $\mathbb{F}_q$. When $b_1 = 0$, then by the second equation we have $-b_2 + b_3 = a_3$, and hence the first equations becomes $a_3y_0 = -b_1 + b_2$, from which we get $y_0 = a_3^{-1}(-b_1 + b_2)$. Therefore, y_0 is a constant in $\mathbb{F}_q$ that determined by $\bar{a}, \bar{b}$. By a similar argument as above, we can prove that the number of solutions of the above system of equations is less than or equal to q^2.

Then we proved that the differential uniformity of Pf^4 is less than or equal to q^2. ∎

Theorem 2: Let Pf^4 be the permutation over $\mathbb{F}_q^4$ in Construction 1. Then the linearity of Pf^4 is less than or equal to q^3.

Proof: Let $\bar{b} = (b_0, b_1, b_2, b_3) \in \mathbb{F}_q^4$ be nonzero vectors, $\bar{a} = (a_0, a_1, a_2, a_3) \in \mathbb{F}_q^4$, and we denote

$$\lambda_{\mathrm{Pf}^4}(\bar{a}, \bar{b}) = \sum_{\bar{x} \in \mathbb{F}_q^4} \chi(\bar{b} \cdot \mathrm{Pf}^4(\bar{x}) - \bar{a} \cdot (\bar{x})).$$

Then we only need to prove $|\lambda_{\mathrm{Pf}^4}(\bar{a}, \bar{b})| \leq q^3$, which is equivalent to

$$|\lambda_{\mathrm{Pf}^4}(\bar{a}, \bar{b}) \cdot \overline{\lambda_{\mathrm{Pf}^4}(\bar{a}, \bar{b})}| \leq q^6.$$

Note that

$$\lambda_{\mathrm{Pf}^4}(\bar{a}, \bar{b}) = \sum_{\bar{x} \in \mathbb{F}_q^4} \chi(\bar{b} \cdot \mathrm{Pf}^2(\bar{x}) - \bar{a} \cdot \mathrm{Pf}^{-2}(\bar{x})),$$

and

$$\begin{aligned}&\bar{b} \cdot \mathrm{Pf}^2(\bar{x}) - \bar{a} \cdot \mathrm{Pf}^{-2}(\bar{x})\\ &= (b_0, b_1, b_2, b_3) \cdot (x_2, x_3, x_0 + x_1x_2 + x_3 + \alpha,\\ &\quad x_1 + x_2x_3 + x_0 + x_1x_2 + x_3 + 2\alpha - (a_0, a_1, a_2, a_3))\\ &\quad \cdot (x_2 - (x_3 - x_0x_1 - x_2 - \alpha)x_0 - x_1 - \alpha,\\ &\quad x_3 - x_0x_1 - x_2 - \alpha, x_0, x_1)\\ &= -a_0x_0^2x_1 + a_1x_0x_1 - a_0x_0x_2 + a_0x_0x_3 + (b_2 + b_3)x_1x_2\\ &\quad +b_3x_2x_3 + c_1x_0 + c_2x_1 + c_3x_2 + c_4x_3 + c_0,\end{aligned}$$

where $c_i \in \mathbb{F}_q, 0 \le i \le 4$ are constants that determined by $\bar{b}$ and $\bar{a}$. Therefore, for $\bar{d} = (d_0, d_1, d_2, d_3) \in \mathbb{F}_q^4$, it holds

$$\begin{aligned}
&\bar{b} \cdot (\mathrm{Pf}^2(\bar{x}) - \mathrm{Pf}^2(\bar{x}+\bar{d})) - \bar{a} \cdot (\mathrm{Pf}^{-2}(\bar{x}) - \mathrm{Pf}^{-2}(\bar{x}+\bar{d})) \\
&= -a_0(d_1x_0^2 + 2d_0x_0x_1 - 2d_0d_1x_0 - d_0^2x_1) \\
&+a_1(d_0x_1 + d_1x_0) - a_0(d_0x_2 + d_2x_0) + a_0(d_0x_3 + d_3x_0) \\
&+(b_2+b_3)(d_1x_2 + d_2x_1) + b_3(d_2x_3 + d_3x_2) + h(\bar{d}) \\
&= -2a_0d_0x_0x_1 - a_0d_1x_0^2 + (2a_0d_0d_1 + a_1d_1 - a_0d_2 \\
&+a_0d_3)x_0 + (a_0d_0^2 + a_1d_0 + (b_2+b_3)d_2)x_1 + (-a_0d_0 \\
&+(b_2+b_3)d_1 + b_3d_3)x_2 + (a_0d_0 + b_3d_2)x_3 + h(\bar{d}),
\end{aligned}$$

where

$$\begin{aligned}
h(\bar{d}) = &-a_0d_0^2d_1 - (a_1d_0d_1 - a_0d_0d_2 + a_0d_0d_3 \\
&+(b_2+b_3)d_1d_2 + b_3d_2d_3) + c_1d_0 \\
&+c_2d_1 + c_3d_2 + c_4d_3. \qquad (2)
\end{aligned}$$

Then we have

$$\begin{aligned}
&|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})|^2 \\
&= \Bigg| \sum_{\bar{x}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot\mathrm{Pf}^2(\bar{x}) - \bar{a}\cdot\mathrm{Pf}^{-2}(\bar{x})) \\
&\quad \cdot \overline{\sum_{\bar{y}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot\mathrm{Pf}^2(\bar{y}) - \bar{a}\cdot\mathrm{Pf}^{-2}(\bar{y}))} \Bigg| \\
&= \Bigg| \sum_{\bar{x}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot\mathrm{Pf}^2(\bar{x}) - \bar{a}\cdot\mathrm{Pf}^{-2}(\bar{x})) \\
&\quad \cdot \sum_{\bar{y}\in\mathbb{F}_q^4} \chi(-\bar{b}\cdot\mathrm{Pf}^2(\bar{y}) + \bar{a}\cdot\mathrm{Pf}^{-2}(\bar{y})) \Bigg| \\
&= \Bigg| \sum_{\bar{x},\bar{d}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot(\mathrm{Pf}^2(\bar{x}) - \mathrm{Pf}^2(\bar{x}-\bar{d})) \\
&\quad +\bar{a}\cdot(\mathrm{Pf}^{-2}(\bar{x}) - \mathrm{Pf}^{-2}(\bar{x}-\bar{d}))) \Bigg| \\
&= \Bigg| \sum_{\bar{d}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot h(\bar{d})) \Bigg| \cdot \Bigg| \sum_{x_0,x_1\in\mathbb{F}_q} \chi(-2a_0d_0x_0x_1 \\
&\quad -a_0d_1x_0^2 + (2a_0d_0d_1 + a_1d_1 - a_0d_2 + a_0d_3)x_0) \\
&\quad \chi((a_0d_0^2 + a_1d_0 + (b_2+b_3)d_2)x_1) \Bigg| \\
&\quad \cdot \Bigg| \sum_{x_2\in\mathbb{F}_q} \chi((-a_0d_0 + (b_2+b_3)d_1 + b_3d_3)x_2) \Bigg| \\
&\quad \cdot \Bigg| \sum_{x_3\in\mathbb{F}_q} \chi((a_0d_0 + b_3d_2)x_3) \Bigg|, \qquad (3)
\end{aligned}$$

and we have the following cases.

Case 1: $a_0 = a_1 = 0$ and $b_2 = b_3 = 0$. Then we have

$$\bar{b}\cdot\mathrm{Pf}^2(\bar{x}) + \bar{a}\cdot\mathrm{Pf}^{-2}(\bar{x}) = b_0x_2 + b_1x_3 + a_2x_0 + a_3x_1.$$

Note that $\bar{b} \neq \bar{0}$, then $b_0 \neq 0$ or $b_1 \neq 0$. Since

$$\sum_{x\in\mathbb{F}_q} \chi(bx) = 0$$

when $b \neq 0$, then we have

$$\begin{aligned}
&\left|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})\right| \\
&= \Bigg| \sum_{\bar{x}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot\mathrm{Pf}^2(\bar{x}) + \bar{a}\cdot\mathrm{Pf}^{-2}(\bar{x})) \Bigg| \\
&= \Bigg| \sum_{x_0\in\mathbb{F}_q} \chi(a_2x_0) \sum_{x_1\in\mathbb{F}_q} \chi(a_3x_1) \sum_{x_2\in\mathbb{F}_q} \chi(b_0x_2) \\
&\quad \sum_{x_3\in\mathbb{F}_q} \chi(b_1x_3) \Bigg| \\
&= 0.
\end{aligned}$$

Case 2: $a_0 = 0$, and $\{a_1, b_2+b_3, b_3\} \neq \{0\}$. Then equation (3) becomes

$$\begin{aligned}
&\left|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})\right|^2 \\
&= \Bigg| \sum_{\bar{d}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot h(\bar{d})) \sum_{x_0\in\mathbb{F}_q} \chi(a_1d_1x_0) \\
&\quad \sum_{x_1\in\mathbb{F}_q} \chi((a_1d_0 + (b_2+b_3)d_2)x_1) \\
&\quad \sum_{x_2\in\mathbb{F}_q} \chi(((b_2+b_3)d_1 + b_3d_3)x_2) \sum_{x_3\in\mathbb{F}_q} \chi(b_3d_2x_3) \Bigg| \\
&= \Bigg| \sum_{\bar{d}\in\mathbb{F}_q^4} \chi(\bar{b}\cdot h(\bar{d})) \Bigg| \cdot \Bigg| \sum_{x_0\in\mathbb{F}_q} \chi(a_1d_1x_0) \Bigg| \cdot \\
&\quad \Bigg| \sum_{x_1\in\mathbb{F}_q} \chi((a_1d_0 + (b_2+b_3)d_2)x_1) \Bigg| \\
&\quad \Bigg| \sum_{x_2\in\mathbb{F}_q} \chi(((b_2+b_3)d_1 + b_3d_3)x_2) \Bigg| \cdot \Bigg| \sum_{x_3\in\mathbb{F}_q} \chi(b_3d_2x_3) \Bigg| \\
&\le |S|q^4,
\end{aligned}$$

where S is the set of $\bar{d} \in \mathbb{F}_q^4$ satisfy the following system of equations

$$\begin{cases}
a_1d_1 = 0 \\
a_1d_0 + (b_2+b_3)d_2 = 0 \\
(b_2+b_3)d_1 + b_3d_3 = 0 \\
b_3d_2 = 0.
\end{cases}$$

Note that

$$\mathrm{rank}\begin{pmatrix}
0 & a_1 & 0 & 0 \\
a_1 & 0 & b_2+b_3 & 0 \\
0 & b_2+b_3 & 0 & b_3 \\
0 & 0 & b_3 & 0
\end{pmatrix} \ge 2,$$

since $\{a_1, b_2+b_3, b_3\} \neq \{0\}$. This means the dimension of the solution space of the above system of equations is less than

or equal to 2. Therefore, $|S| \le q^2$, and hence

$$\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})^2 \le |S|q^4 \le q^6.$$

Case 3: $a_0 \neq 0$. Then equation (3) becomes

$$\begin{aligned}
&\left|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})\right|^2\\
&=\Bigg|\sum_{\bar{d}\in\mathbb{F}_q^4}\chi(\bar{b}\cdot h(\bar{d}))\sum_{x_0,x_1\in\mathbb{F}_q}\chi(-2a_0d_0x_0x_1-a_0d_1x_0^2\\
&\quad+(2a_0d_0d_1+a_1d_1-a_0d_2+a_0d_3)x_0)\\
&\quad\chi((a_0d_0^2+a_1d_0+(b_2+b_3)d_2)x_1)\\
&\quad\sum_{x_2\in\mathbb{F}_q}\chi((-a_0d_0+(b_2+b_3)d_1+b_3d_3)x_2\\
&\quad\sum_{x_3\in\mathbb{F}_q}\chi((a_0d_0+b_3d_2)x_3)\Bigg|\\
&=q^2\Big|\sum_{\bar{d}\in S}\chi(\bar{b}\cdot h(\bar{d}))\Big|\cdot\Big|\sum_{x_0,x_1\in\mathbb{F}_q}\chi(-2a_0d_0x_0x_1-a_0d_1x_0^2\\
&\quad+(2a_0d_0d_1+a_1d_1-a_0d_2+a_0d_3)x_0)\\
&\quad\chi((a_0d_0^2+a_1d_0+(b_2+b_3)d_2)x_1)\Big|,
\end{aligned}$$

where S is the set of $\bar{d}\in\mathbb{F}_q^4$ satisfy the following system of equations

$$\begin{cases} a_0d_0+(b_2+b_3)d_1+b_3d_3=0\\ a_0d_0+b_3d_2=0. \end{cases} \tag{4}$$

Case 3.1: $b_3 \neq 0$ or $b_2+b_3\neq 0$. Then

$$\operatorname{rank}\begin{pmatrix} a_0b_2+b_3 & 0 & b_3\\ a_0 & 0 & b_3\ 0 \end{pmatrix}=2.$$

Therefore, the dimension of the solution space of the above system of equations is equal to 2. Thus $|S|\le q^2$ and hence

$$|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})|^2 \le q^2|S|q^2 \le q^6.$$

Case 3.2: $b_2=b_3=0$. Then we get $d_0=0$ from equation systems (4). Then equation (3) becomes

$$\begin{aligned}
&|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})|^2\\
&=q^2\Bigg|\sum_{d_1,d_2,d_3\in\mathbb{F}_q}\chi(\bar{b}\cdot h(\bar{d}))\\
&\quad\sum_{x_0,x_1\in\mathbb{F}_q}\chi(-a_0d_1x_0^2+(a_1d_1-a_0d_2+a_0d_3)x_0)\Bigg|\\
&=q^3\Bigg|\sum_{d_1,d_2,d_3\in\mathbb{F}_q}\chi(\bar{b}\cdot h(\bar{d}))\\
&\quad\sum_{x_0\in\mathbb{F}_q}\chi(-a_0d_1x_0^2+(a_1d_1-a_0d_2+a_0d_3)x_0)\Bigg|.
\end{aligned}$$

Note that when $b_2=b_3=0$ and $d_0=0$, from equation (2), we have $h(\bar{d})=c_2d_1+c_3d_2+c_4d_3$. Thus

$$\begin{aligned}
&|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})|^2\\
&=q^3\Bigg|\sum_{d_1,d_2,d_3\in\mathbb{F}_q}\chi(b_1c_2d_1+b_2c_3d_2+b_3c_4d_3)\\
&\quad\sum_{x_0\in\mathbb{F}_q}\chi(-a_0d_1x_0^2+(a_1d_1-a_0d_2+a_0d_3)x_0)\Bigg|\\
&=q^3\sum_{x_0\in\mathbb{F}_q}\Bigg|\sum_{d_1\in\mathbb{F}_q}\chi((b_1c_2-a_0x_0^2+a_1x_0)d_1)\Bigg|\\
&\quad\Bigg|\sum_{d_2\in\mathbb{F}_q}\chi((b_2c_3-a_0x_0)d_2)\Bigg|\Bigg|\sum_{d_3\in\mathbb{F}_q}\chi((b_3c_4-a_0x_0)d_3)\Bigg|.
\end{aligned}$$

Note that there is at most one x_0 such that

$$\begin{cases} b_1c_2-a_0x_0^2+a_1x_0=0\\ b_2c_3-a_0x_0=0\\ b_3c_4-a_0x_0=0. \end{cases}$$

Furthermore, we have

$$|\lambda_{\mathrm{Pf}^4}(\bar{a},\bar{b})|^2 \le q^6.$$

Then we complete the proof. ∎

IV. Description of YuX

As an application of the constructed nonlinear function with low multiplication depth, we propose the design of the block cipher named YuX. This cipher is based on the use of Pf^4 over $\mathbb{F}_q^4$, where q is a prime number p or a power of 2. In the following, we give a detailed description of YuX, including the encryption, decryption, key generation, and round constant generation algorithms. We also give two instances named $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$, which are block ciphers over $\mathbb{F}_{2^n}^{16}$ and $\mathbb{F}_p^{16}$ respectively.

A. Cipher Description

YuX is a block cipher over $\mathbb{F}_q^{16}$ based on the SPN structure. The encryption starts with a key whitening, followed by a number of round transformations, as shown in Fig. 2. The $(i+1)$-th round transformation is

$$Round_{i+1}(x_0^i,\ldots,x_{15}^i)=AK_{rk^{i+1}}\circ LP\circ SL(x_0^i,\ldots,x_{15}^i),$$

where the "SL", "LP", and "AK" are "S-box Layer", "Linear Layer", and "Adding Round-key" operations.

1) S-Box Layer:

$$\begin{aligned}
SL(x_0,\ldots,x_{15})=(&\mathrm{S}(x_0,\ldots,x_3),\mathrm{S}(x_4,\ldots,x_7),\\
&\mathrm{S}(x_8,\ldots,x_{11}),\mathrm{S}(x_{12},\ldots,x_{15})),
\end{aligned}$$

which has 4 S-boxes, and each S-box is over $\mathbb{F}_q^4$. Its inverse operation is

$$\begin{aligned}
SL_{inv}(x_0,\ldots,x_{15})=(&\mathrm{S}^{-1}(x_0,\ldots,x_3),\mathrm{S}^{-1}(x_4,\ldots,x_7),\\
&\mathrm{S}^{-1}(x_8,\ldots,x_{11}),\mathrm{S}^{-1}(x_{12},\ldots,x_{15})),
\end{aligned}$$

where S is a permutation over $\mathbb{F}_q^4$ and S^{-1} is the compositional inverse of S.

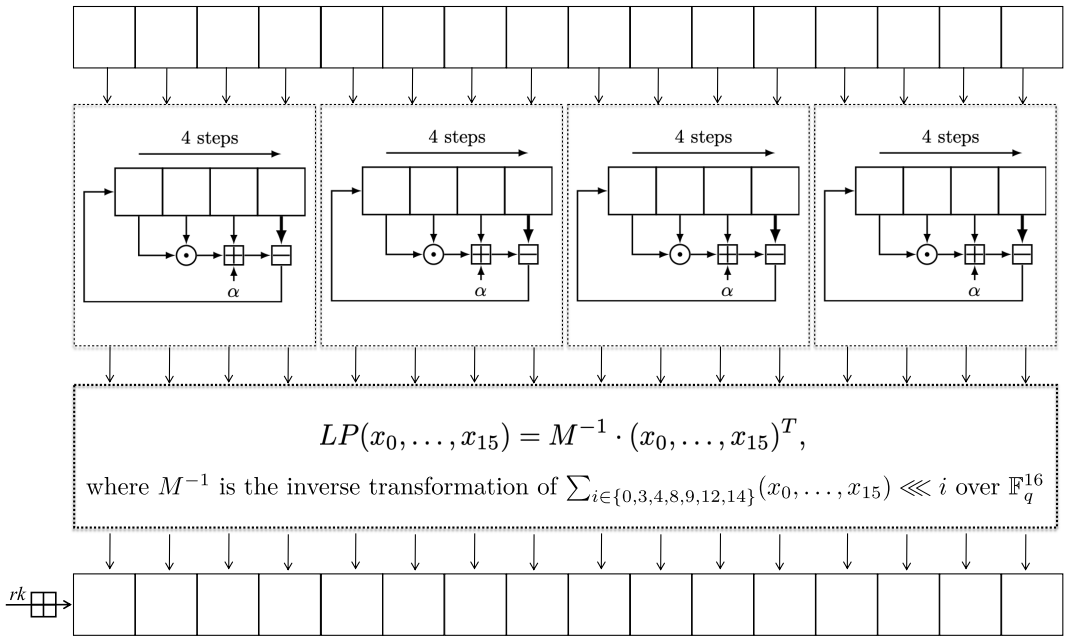


Fig. 2. One round encryption of YuX.

2) S and S^{-1}: The S-box is a permutation over $\mathbb{F}_q^4$ of Construction 1. Define

$$S(x_0,x_1,x_2,x_3) = Pf^{-4}(x_0,x_1,x_2,x_3),$$

and

$$S^{-1}(x_0,x_1,x_2,x_3) = Pf^{4}(x_0,x_1,x_2,x_3).$$

3) Linear Layer: Let M be the matrix over $\mathbb{F}_q^{16}$ such that

$$M\cdot(x_0,\ldots,x_{15})^T = \sum_{i\in\{0,3,4,8,9,12,14\}} (x_0,\ldots,x_{15}) \lll i,$$

where the rotations are based on words.[2] Let M^{-1} be the inverse of M over $\mathbb{F}_q^{16}$. Then the linear layer of YuX is defined as

$$LP(x_0,\ldots,x_{15}) = M^{-1}\cdot(x_0,\ldots,x_{15})^T$$

and

$$LP_{inv}(x_0,\ldots,x_{15}) = \sum_{i\in\{0,3,4,8,9,12,14\}} (x_0,\ldots,x_{15}) \lll i,$$

4) Adding Round-Key:

$$AK_{rk^{i+1}}(x_0^i,\ldots,x_{15}^i) = (x_0^i + rk_0^i,\ldots,x_{15}^i + rk_{15}^i).$$

The key schedule algorithm includes the operations of rotation, S-box, and XOR with Round Constant, as shown in Algorithm 1, and the Round Constant generation is shown in Algorithm 2.

Algorithm 1 Key Schedule Algorithm: YuX-KS(Key)

Input: $Key = (key_0,\ldots,key_{15})$,
$rc = (rc^0,\ldots,rc^{r-1})$, for $rc^i = (rc_0^i, rc_1^i,\ldots,rc_{15}^i)$
Output: $RoundKey = \text{YuX-KS}(Key, rc)$

$rk^0 = Key$
$X_i = (key_i, key_{i+1}, key_{i+2}, key_{i+3}), i \in \{0,\ldots,3\}$
for $i = 1\ldots r$ **do**
  $rk^i = ()$
  for $j = 0\ldots3$ **do**
    $X_4 = X_0 + S((X_1+X_2+X_3) \lll 3) + (rc_{4j}^{i-1},\ldots,rc_{4j+3}^{i-1})$
    $rk^i = rk^i || X_4$
    $X_0 = X_1,\ X_1 = X_2,\ X_2 = X_3, X_3 = X_4$
  end for
end for
return $rk = (rk^0, rk^1,\ldots,rk^r)$

B. Specifications of YuX

In the following, we present two distinct variants of the YuX: one operating over finite fields $\mathbb{F}_{2^n}$ as Yu_2X, and the other over prime fields $\mathbb{F}_p$ denoted as Yu_pX. We give some specifications of Yu_2X and Yu_pX in Table II.

1) Yu_2X-n: The block cipher is constructed over $\mathbb{F}_{2^n}^{16}$, where $n \geq 8$. More specifically, within the encryption algorithm, the S-box is the permutation Pf^{-4} over $\mathbb{F}_{2^n}^4$ and the LP operation of Yu_2X can be mathematically expressed as:

$$LP(x_0,\ldots,x_{15}) = \oplus_{i\in\{1,2,3,5,6,7,8,12,13,14,15\}} (x_0,\ldots,x_{15}) \lll i.$$

The round number of Yu_2X is not less than 12 when configured for 128-bit security.

2) Yu_pX-p: The block cipher is constructed over $\mathbb{F}_p^{16}$, where $log_2(p-1) \geq 16$. More specifically, the S-box is the permutation Pf^{-4} over $\mathbb{F}_p^4$ and and the LP operation in the encryption algorithm of Yu_pX is

$$LP(x_0,\ldots,x_{15}) = M^{-1}\cdot(x_0,\ldots,x_{15})^T.$$

[2] $(x_0,\ldots,x_{15}) \lll 1 = (x_1,\ldots,x_{15},x_0)$.

Algorithm 2 Round Constant: YuX-ConsGen()

Input: $rc = ()$
Output: $RoundConstant$ = YuX-ConsGen()
for $i = 0 \ldots r-1$ **do**
  $rc^i = ()$
  for $j = 0 \ldots 3$ **do**
    $id = 16 * i + 4 * j$
    $(c_0, c_1, c_2, c_3) = \mathrm{S}(id+1, id+2, id+3, id+4)$
    $rc^i = rc^i || (c_0, c_1, c_2, c_3)$
  end for
end for
return $rc = (rc^0, rc^1, \ldots, rc^{r-1})$

Algorithm 3 Encryption Algorithm of YuX

Input: $PlainText = (x_0^0, x_1^0, \ldots, x_{15}^0)$,
$rk = (rk^0, rk^1, \ldots, rk^r)$, for $rk^i = (rk_0^i, \ldots, rk_{15}^i)$
Output: $CipherText$ = YuX-Enc($PlainText, rk$)
$(x_0^0, \ldots, x_{15}^0) = (x_0^0 + rk_0^0, \ldots, x_{15}^0 + rk_{15}^0)$
for $i = 1 \ldots r-1$ **do**
  // S-box Layer
  $(x_0^i, \ldots, x_{15}^i) = \mathrm{S}(x_0^{i-1}, \ldots, x_3^{i-1}) || \mathrm{S}(x_4^{i-1}, \ldots, x_7^{i-1}) || \mathrm{S}(x_8^{i-1}, \ldots, x_{11}^{i-1}) || \mathrm{S}(x_{12}^{i-1}, \ldots, x_{15}^{i-1})$
  // Linear Layer
  $(x_0^i, \ldots, x_{15}^i) = LP(x_0^i, \ldots, x_{15}^i)$
  // Add RoundKey
  $(x_0^i, \ldots, x_{15}^i) = (x_0^i + rk_0^i, \ldots, x_{15}^i + rk_{15}^i)$
end for
$(x_0^r, \ldots, x_{15}^r) = \mathrm{S}(x_0^{r-1}, \ldots, x_3^{r-1}) || \mathrm{S}(x_4^{r-1}, \ldots, x_7^{r-1}) || \mathrm{S}(x_8^{r-1}, \ldots, x_{11}^{r-1}) || \mathrm{S}(x_{12}^{r-1}, \ldots, x_{15}^{r-1})$
$(x_0^r, \ldots, x_{15}^r) = (x_0^r + rk_0^r, \ldots, x_{15}^r + rk_{15}^r)$
return $(x_0^r, \ldots, x_{15}^r)$

The matrix M^{-1} is identifiable as a circulant matrix, as delineated in C, which enables an alternative representation of the LP operation as

$$LP(x_0, \ldots, x_{15}) = [v_p \cdot ((x_0, \ldots, x_{15}) \lll i) : i \in [0..15]],$$

where

$$v_p = (\frac{4}{7}, \frac{5}{21}, \frac{5}{21}, \frac{5}{21}, \frac{4}{7}, \frac{5}{21}, \frac{5}{21}, \frac{5}{21}, -\frac{3}{7}, -\frac{16}{21}, -\frac{16}{21}, -\frac{16}{21}, -\frac{3}{7}, \frac{5}{21}, \frac{5}{21}, \frac{5}{21}),$$

is a vector over $\mathbb{F}_p^{16}$, and $\frac{b}{a}$ representing $b \cdot a^{-1}$ over $\mathbb{F}_p$. Illustratively, when the value of p is 65537, the corresponding v_p is defined as:

$$v_p = (9363, 53054, 53054, 53054, 9363, 53054, 53054, 53054, 9362, 53053, 53053, 53053, 9362, 53054, 53054, 53054).$$

For the security setting of 118-bit, the round number of $\mathrm{Yu_pX}$ is not less than 9.

V. Design Rationale

The primary objective is to devise a block cipher possessing a decryption circuit conducive to the implementation of fully homomorphic encryption, with a concurrent emphasis on maximizing operational efficiency.

A. The Operations Based

Based on our experimental results, the efficiency of Fully Homomorphic Encryption (FHE) implementations is primarily influenced by two factors: the complexity of multiplication and the running time of basic operations. Multiplication complexity refers to the number of computationally expensive operations such as homomorphic multiplication and automorphism. The running time of basic operations is mainly determined by the value of $\phi(m)$ and the current level of the leveled FHE scheme. As the multiplication depth increases by one, the corresponding increase in the level demands a higher amount of time for the basic operations to execute. Hence, the multiplication depth of the cipher significantly impacts the speed of homomorphic basic operations. In general, the design of a homomorphic encryption scheme should aim to minimize the number of multiplications and automorphisms while keeping the depth as low as possible.

1) Bit-Based or Field-Based: The choice of using state-sliced over bit-sliced in the FHE implementation of YuX was based on several factors. According to the BGV scheme supporting SIMD operations, $\Phi_m(x)$ can be split into r irreducible polynomial with a degree of $d = \phi(m)/r$, and r is the number of slots. Although bit-sliced block ciphers typically have smaller $\phi(m)$, resulting in faster homomorphic operations at the same depth, they require encoding each bit into a separate FHE ciphertext, leading to higher communication overhead and wasting plaintext space. Additionally, block ciphers over $\mathbb{F}_2$ usually require a large number of AND gates, leading to longer total running times in FHE implementations. Therefore, the design of YuX chose field-based operations over $\mathbb{F}_q$, where q is a prime p or $q = 2^n$ for improving FHE evaluation efficiency and block cipher throughput.

B. Selection of Structure

The Feistel/generalized Feistel and the SPN structure are two main structures in the design of block ciphers. Implementing a Feistel-based cipher's decryption algorithm is easy and almost the same as the encryption algorithm, but a Feistel-based cipher usually has a rather large number of rounds. An SPN-based cipher usually has fewer rounds than a Feistel-based cipher. In contrast, the decryption of an SPN-based cipher has to implement the inverse of the nonlinear and linear components in the cipher.

The multiplication depth of the cipher plays an essential role in the FHE implementation. Then we choose SPN as the structure of YuX since it has few rounds and leads to a relatively low multiplication depth due to a nonlinear function having a lower multiplication depth.

C. Design of S^{-1} and S

The nonlinear function provides "confusion" for a cipher. The main aim of our construction for the S-box of YuX is that

1) The multiplication depth of its inverse (used in the decryption algorithm) is less than or equal to 2.

TABLE II
SPECIFICATIONS OF $\mathsf{Yu_2X}$ AND $\mathsf{Yu_pX}$

Ciphers	Block	Defining Polynomial	Constant	Security level	Rounds
$\mathsf{Yu_2X}$-8	$\mathbb{F}_{2^8}^{16}$	$x^8+x^4+x^3+x+1$	205	128 bit	12
$\mathsf{Yu_2X}$-16	$\mathbb{F}_{2^{16}}^{16}$	$x^{16}+x^{12}+x^3+x+1$	205	128 bit	12* 14
$\mathsf{Yu_pX}$-65537	$\mathbb{F}_{65537}^{16}$	—	205	118 bit* 128 bit	9* 14

* The recommended number of rounds in fully homomorphic application scenarios.

TABLE III
COSTS OF HOMOMORPHIC OPERATIONS

Operation	Noise (Levels)	Time consume (Magnitude)
pt-ct Add	negligible (0)	cheap(1)
ct-ct Add	negligible (0)	cheap(1)
pt-ct Mul	moderate (0.5)	cheap ($7*10^1$)
ct-ct Mul	expensive (1)	expensive($1*10^3$)
Automorphism	moderate (0.5)	expensive($4*10^2$)

Magnitude: the multiple of the running time of addition.

2) Lower the number of nonlinear operations and automorphisms used in constructing S-boxes.
3) The differential uniformity and linearity can be characterized.

According to Table III, an automorphism increases 0.5 multiplication depth. Thus we avoid using automorphisms in the construction of S^{-1}. Based on the above consideration, we design S^{-1} and S based on nonlinear feedback resisters over finite fields, and the differential uniformity and linearity are also characterized in Theorem 1 and Theorem 2 respectively.

D. Design of Linear Layer

Linear mapping provides "diffusion" for a cipher. The main aim of our construction for the linear mapping of YuX is that

1) It should be an MDS mapping over $(\mathbb{F}_q^4)^4$.
2) There are no operations of multiplying by a constant in $\mathbb{F}_q$ in the linear mapping used in the decryption algorithm.

According to the construction of S and S^{-1}, they can be viewed as S-boxes over $\mathbb{F}_q^4$. Compared with nonlinear operations, linear operations are almost free in the FHE implementation. Then we choose MDS mappings over $(\mathbb{F}_q^4)^4$, since they can provide the best diffusion over four S-boxes over $\mathbb{F}_q^4$.

According to Table III, multiplying by a constant in $\mathbb{F}_q$ (pt-ct Mul) increases 0.5 multiplication depth and is 70 times slower than addition. Then we want to design an MDS matrix only with Additions in $\mathbb{F}_q$. Then we construct the MDS mapping via vector rotations and Additions, and we choose

$$LP_{inv}(x_0,\ldots,x_{15}) = \sum_{i\in\{0,3,4,8,9,12,14\}} (x_0,\ldots,x_{15}) \lll i$$

in the decryption algorithms, and its inverse over $\mathbb{F}_{2^n}^{16}$ and $\mathbb{F}_p^{16}$ are used in the encryption of $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$ respectively.

VI. SECURITY ANALYSIS OF YUX

In this section, we evaluate the security of YuX against some main cryptanalysis methods. YuX demonstrates applicability within fully homomorphic scenarios, where ciphertexts undergo homomorphic decryption rather than direct decryption. Therefore, chosen-ciphertext attacks are out of scope since the ciphertext remains safeguarded by the homomorphic algorithm.

A. Differential Attack and Linear Attack

Differential [41] and linear [42] cryptanalysis can be considered the cornerstone of modern cryptanalysis techniques for symmetric ciphers. Resistances against these two attacks are regarded as the baseline in the design of new primitives.

According to the proof Theorem 1 and Theorem 2 in Sect. III-B, as for YuX, the differential uniformity and the linearity of S is

$$\Delta(S) \le q^2, \mathcal{L}(S) \le q^3.$$

This means the maximal differential probability and the maximal linear probability of S is

$$DP^S \le \frac{q^2}{q^4} = q^{-2}, LP^S \le \left(\frac{q^3}{q^4}\right)^2 = q^{-2}.$$

Note that the linear layer

$$LP(x_0,\ldots,x_{15})$$

are MDS mappings over $(\mathbb{F}_{2^n}^4)^4$ and $(\mathbb{F}_p^4)^4$. Then the k rounds' minimum active S-boxes (S) are listed in Table IV.

Therefore, the security bound (128bit) against differential attack and linear attack of $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$ is 4 rounds and 2 rounds respectively.

B. Impossible Differential Attack and Zero Correlation Attack

Impossible differential cryptanalysis was introduced independently to analyze the security of `Skipjack` [43] and `DEAL` [44]. Unlike differential cryptanalysis, impossible differential cryptanalysis tries to find a differential characteristic with zero probability.

Using the SAT method, we find that the longest impossible differential of $\mathsf{Yu_2X}$ is 2 rounds (two S-box layer),

TABLE IV
NUMBER OF ACTIVE DIFFERENTIAL/LINEAR S-BOXES AND THE PROBABILITIES OF DCS AND LTS

Round	1	2	3	4	5	6
#DS, #LS	1	5	6	10	11	15
$\mathsf{Yu_2X}$: Probability	2^{-2n}	2^{-10n}	2^{-12n}	2^{-20n}	2^{-22n}	2^{-30n}
$\mathsf{Yu_pX}$: Probability	p^{-2}	p^{-10}	p^{-12}	p^{-20}	p^{-22}	p^{-30}
$\mathsf{Yu_2X}$-8: Probability	2^{-16}	2^{-80}	2^{-96}	2^{-160}	2^{-176}	2^{-240}
$\mathsf{Yu_pX}$(65537): Probability	2^{-32}	2^{-160}	2^{-192}	2^{-320}	2^{-352}	2^{-480}

and there is no 3-round impossible differential with one S-box active. Some of our research results are shown in the following.

$$(\mathcal{A}^8 0^8 0^8 0^8\ 0^{32}\ 0^{32}\ 0^{32}) \nrightarrow (\mathcal{A}^8 0^8 0^8 0^8\ 0^{32}\ 0^{32}\ 0^{32})$$
$$(\mathcal{A}^8 \mathcal{A}^8 0^8 0^8\ 0^{32}\ 0^{32}\ 0^{32}) \nrightarrow (0^{32}\ \mathcal{A}^8 0^8 0^8 0^8\ 0^{32}\ 0^{32})$$
$$(\mathcal{A}^8 \mathcal{A}^8 \mathcal{A}^8 \mathcal{A}^8\ 0^{32}\ 0^{32}\ 0^{32}) \nrightarrow (0^{32}\ \mathcal{A}^8 \mathcal{A}^8 \mathcal{A}^8 \mathcal{A}^8\ 0^{32}\ 0^{32})$$

Zero-correlation linear cryptanalysis was introduced in [45]. Let Γ_{in} and Γ_{out} be the input and output mask, respectively. Zero-correlation attacks exploit the pair $(\Gamma_{in}, \Gamma_{out})$ with correlation exactly zero. The main technique to derive the zero-correlation linear hull is similar to the impossible differential. We find that the longest zero-correlation linear hull of $\mathsf{Yu_2X}$ is 2 rounds, and there is no 3-round zero-correlation linear hull with one S-box active. The above 2-round impossible differential distinguishers are also 2-round zero-correlation distinguishers.

As for $\mathsf{Yu_pX}$, it can be checked that the similar distinguishers above are also 2-round impossible differential distinguishers and zero correlation attack distinguishers of $\mathsf{Yu_pX}$. We also claim that $\mathsf{Yu_pX}$ also does not have 3-round impossible differential distinguishers and zero correlation attack distinguishers.

C. Integral Attack

The integral attack is one of the chosen plaintext attacks considering the propagation of sums of (many) values. Division property, proposed as the generalization of the integral property [46], has improved many previous integral distinguishers for block ciphers.

1) Division Property of $\mathsf{Yu_2X}$: For $n = 8$: The bit-based division property is an integral characteristic of block cipher on the bit level. We searched integral distinguishers for $\mathsf{Yu_2X}$ with bit-based division property. Since the S-box of $\mathsf{Yu_2X}$ is operated over $\mathbb{F}_{2^8}^4$, using the method in [47] to accurately model the division trails of the S-box by several linear inequalities will generate a huge number of inequalities that cannot be optimized. Therefore, we make a compromise way to model the propagation characteristic of the S-box. We computed the ANF of Pf in the S-box (as shown in Appendix A), which means that the S-box can be decomposed into a sequence of basic bit-wise COPY, AND, and XOR operations. Then, we model bit-wise operations by several linear inequalities and use the MILP (Mixed Integer Linear Programming) optimizer Gurobi[3] to detect the integral distinguisher.

Let $\mathcal{A}$, C, $\mathcal{B}$, and $\mathcal{U}$ represent ACTIVE, CONSTANT, BALANCE, and UNKNOWN bit, respectively. The longest integral distinguisher we found for $\mathsf{Yu_2X}$ is 6 rounds and is attained by utilizing a total of 2^{127} chosen plaintexts. Some distinguishers are listed as follows,

$$\begin{aligned}
&(\left[C^1\mathcal{A}^{15}\right]\mathcal{A}^{16}, \mathcal{A}^{32}, \mathcal{A}^{32}, \mathcal{A}^{32}) \xrightarrow{6R} (\mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16})\\
&(\mathcal{A}^{32}, \left[C^1\mathcal{A}^{15}\right]\mathcal{A}^{16}, \mathcal{A}^{32}, \mathcal{A}^{32}) \xrightarrow{6R} (\mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16})\\
&(\mathcal{A}^{32}, \mathcal{A}^{32}, \left[C^1\mathcal{A}^{15}\right]\mathcal{A}^{16}, \mathcal{A}^{32}) \xrightarrow{6R} (\mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16})\\
&(\mathcal{A}^{32}, \mathcal{A}^{32}, \mathcal{A}^{32}, \left[C^1\mathcal{A}^{15}\right]\mathcal{A}^{16}) \xrightarrow{6R} (\mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}, \mathcal{U}^{16}\mathcal{B}^{16}),
\end{aligned}$$

where $\left[C^1\mathcal{A}^{15}\right]$ represents that there is one-bit CONSTANT in an arbitrary position of these 16 bits and the other 15 bits are ACTIVE.

2) An Estimation of the Division Property of YuX on Big Fields: For $f(x) \in \mathbb{F}_q[x]$, $\sum_{a\in\mathbb{F}_q} f(a) = 0$ when $\deg(f) \le q-2$. Then we want to estimate the round number r such that the degree of the r-round output can reach $(q-1)\cdot n_r$ for $q = 2^{16}$ and 65537, where n_r is the number of active words. Since the items grow exponentially as the round increase, it is impossible to compute the exact ANF for the larger round number. Then we estimate the bound according to the following two steps:

1) Computing ANF in the polynomial ring $\mathbb{F}_q[x_0,\ldots,x_{15}]/\langle x_0^2 - x_0, \ldots, x_{15}^2 - x_{15}\rangle$. After the third and the second round, $\prod_{i=0}^{15} x_i$ would appear in the ANF of the 16 output words for $q = 2^{16}$ and $q = 65537$ respectively.
2) Computing the degree of the output words in the polynomial ring $\mathbb{F}_q[x]/\langle x^q - x\rangle$. For the input $(a_0x + b_0, \ldots, a_{15}x + b_{15})$, where $a_i, b_i, 0 \le i \le 15$ are random constants, all the output words after the S-box layer are polynomials with degree $q-1$ after 7 rounds and 6 rounds for $q = 2^{16}$ and $q = 65537$ respectively.

According to the above two experiments, we estimate that for the encryption of $\mathsf{Yu_2X}$−16 and $\mathsf{Yu_pX}$−65537, the division distinguisher can not exceed 10 and 8 rounds respectively. This is because the monomial $\prod_{i=0}^{15} x_i^{q-1}$ would appear in the computation process of the tenth and eighth round output words. The following is a 6-round distinguisher of $\mathsf{Yu_2X}$−16 and $\mathsf{Yu_pX}$−65537 with q plaintext complexity.

$$(\mathcal{A}^q, C, \ldots, C) \xrightarrow{6R} (\mathcal{U}^q\mathcal{U}^q\mathcal{B}^q, \mathcal{B}^q, \ldots, \mathcal{U}^q\mathcal{U}^q\mathcal{B}^q, \mathcal{B}^q).$$

[3]Gurobi Optimization, https://www.gurobi.com/.

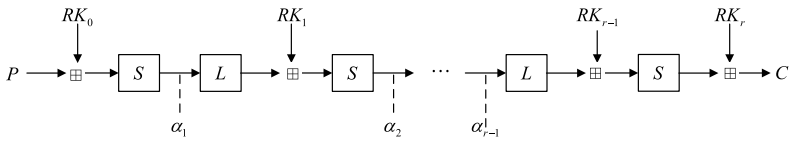


Fig. 3. Intermediate variables in Gröbner basis attack.

D. Algebraic Attack

1) Gröbner Basis Attack: We follow the steps in [27] to perform a Gröbner Basis Attack. Firstly, we set up the equation system for YuX family.

For the S-box, let the input and output be $x = (x_0, x_1, x_2, x_3) \in \mathbb{F}_q^4$ and $y = (y_0, y_1, y_2, y_3) \in \mathbb{F}_q^4$, then $S(x) = y$ is equivalent to the following 4 quadratic equations on the finite field $\mathbb{F}_q$, where q = prime p or 2^n and module addition + and minus − both denote $\oplus$ for $\mathsf{Yu_2X}$.

$$\begin{aligned} y_3 &= x_0 \cdot x_1 + x_2 + \alpha - x_3, \\ y_2 &= y_3 \cdot x_0 + x_2 + \alpha - x_1, \\ y_1 &= y_2 \cdot y_3 + x_0 + \alpha - x_1, \\ y_0 &= y_1 \cdot y_2 + y_3 + \alpha - x_0. \end{aligned}$$

There are 4 parallel S-boxes in the S-box layer. We denote the S-box layer as $SL(X, Y) = 0, X, Y \in \mathbb{F}_q^{16}$, which corresponds to 16 quadratic equations on the finite field $\mathbb{F}_q$. Denote the state between the S-box layer SL and linear layer LP in round i as $\alpha_i \in \mathbb{F}_q^{16}$, for $i = 1, 2, \ldots, r-1$ (see Fig. 3). Let the round key for whitening and each round denote as $RK_i \in \mathbb{F}_q^{16}$, for $i = 0, 1, \ldots, r$. For each plaintext and ciphertext pair (P, C), we have the following $16r$ quadratic equations on the finite field $\mathbb{F}_q$,

$$\begin{aligned} SL(P + RK_0, \alpha_1) &= 0, \\ SL(LP(\alpha_i) + RK_i, \alpha_{i+1}) &= 0, i = 1, \ldots, r-2 \\ SL(LP(\alpha_{r-1}) + RK_{r-1}, RK_r + C) &= 0 \end{aligned}$$

For the key schedule, set $RK_i = (RK_{i,0}, RK_{i,1}, RK_{i,2}, RK_{i,3})$, for $i = 0, 1, \ldots, r$. Let

$$RK_{0,i} = [k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}], i = 0, 1, 2, 3.$$

Denotes $LK(x_0, x_1, x_2) = [x_0 + x_1 + x_2] \lll 3$ where $x_i \in \mathbb{F}_q^4$, for $i = 0, 1, 2$, then we have the following $16r$ quadratic equations on the finite field $\mathbb{F}_q$,

$$\begin{cases} RK_{i,0} = S(LK(RK_{i-1,1}, RK_{i-1,2}, RK_{i-1,3})) \\ \quad + RK_{i-1,0} + RC, \\ RK_{i,1} = S(LK(RK_{i-1,2}, RK_{i-1,3}, RK_{i,0})) + RK_{i-1,1} \\ \quad + RC, \\ RK_{i,2} = S(LK(RK_{i-1,3}, RK_{i,0}, RK_{i,1})) + RK_{i-1,2} \\ \quad + RC, \\ RK_{i,3} = S(LK(RK_{i,0}, RK_{i,1}, RK_{i,2})) + RK_{i-1,3} + RC, \end{cases}$$

for $i = 1, 2, \ldots, r$.

In total, the above description of the cipher amounts to $n_e = 32r$ quadratic equations on the finite field of $\mathbb{F}_q$ in $n_v = 32r$ variables of $RK_0, \ldots, RK_r, \alpha_1, \alpha_{r-1}$.

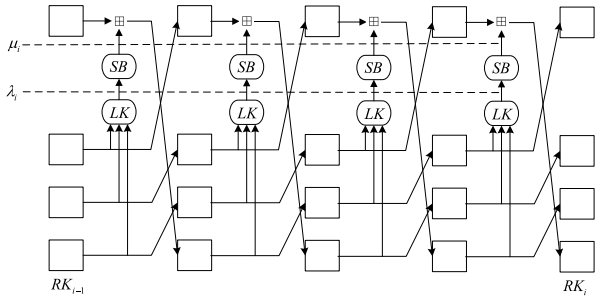


Fig. 4. Intermediate variables of the key schedule in algebraic attack by linearization.

Next, it needs to compute the Gröbner basis to the equation system. We assume the system behaves like regular sequences for $n_e = n_v$. Then the complexity to compute the Gröbner Basis is $O\left(\binom{n_v + D_{reg}}{D_{reg}}^{\omega}\right)$, where $2 \le \omega < 3$, and D_{reg} is the degree of regularity. The degree of regularity for regular sequences is given by $D_{reg} = 1 + \sum_{i=1}^{n_e}(d_i - 1)$, where d_i is the degree of each equation in the system. Here, we take $\omega = 2$ to bound the lower complexity, and its results in

$$\binom{32 \times r + 32 \times r + 1}{32 \times r + 1}^2$$

for r rounds. When $r = 2$, it is approximately $2^{250.3}$ and achieves the security bound.

Since the complexity for calculating the Gröbner basis is over the 128-bit security for the two instances, we omit the next steps in the Gröbner basis attack.

2) Algebraic Attack by Linearization: For methods to solve equations, the linearization technique is common, which is to treat each different high-degree monomial in the equations as an independent new variable. Then the equation system becomes linear and can be solved by Gaussian elimination.

The challenge is hence to collect sufficiently many equations, whose scale always equals the number of variables in a very conservative way. To control the expansion of variables, we need to focus on the algebraic degree and the expression of multiples. Therefore, we rewrite the equation system on the finite field as follows.

For the key schedule, set $\mu_0 = (\mu_{0,0}, \mu_{0,1}, \mu_{0,2}, \mu_{0,3}) \in \mathbb{F}_q^{16}$. Let

$$\mu_{0,i} = [K_{4i}, K_{4i+1}, K_{4i+2}, K_{4i+3}], i = 0, 1, 2, 3.$$

Let the input and output of the S-box layer for round i be $\lambda_i = (\lambda_{i,0}, \lambda_{i,1}, \lambda_{i,2}, \lambda_{i,3}) \in \mathbb{F}_q^{16}$ and $\mu_i = (\mu_{i,0}, \mu_{i,1}, \mu_{i,2}, \mu_{i,3}) \in \mathbb{F}_q^{16}$ for $i = 1, 2, \ldots, r$(see Fig. 4),

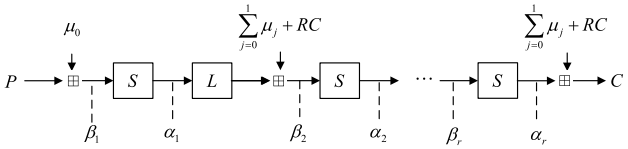


Fig. 5. Intermediate variables of the round function in algebraic attack by linearization.

we have

$$\lambda_{i,0} = LK(\sum_{j=0}^{i-1}\mu_{j,1} + RC, \sum_{j=0}^{i-1}\mu_{j,2} + RC, \sum_{j=0}^{i-1}\mu_{j,3} + RC),$$
$$S(\lambda_{i,0},\mu_{i,0}) = 0,$$
$$\lambda_{i,1} = LK(\sum_{j=0}^{i-1}\mu_{j,2} + RC, \sum_{j=0}^{i-1}\mu_{j,3} + RC, \sum_{j=0}^{i}\mu_{j,0} + RC),$$
$$S(\lambda_{i,1},\mu_{i,1}) = 0,$$
$$\lambda_{i,2} = LK(\sum_{j=0}^{i-1}\mu_{j,3} + RC, \sum_{j=0}^{i}\mu_{j,0} + RC, \sum_{j=0}^{i}\mu_{j,1} + RC),$$
$$S(\lambda_{i,2},\mu_{i,2}) = 0,$$
$$\lambda_{i,3} = LK(\sum_{j=0}^{i}\mu_{j,0} + RC, \sum_{j=0}^{i}\mu_{j,1} + RC, \sum_{j=0}^{i}\mu_{j,2} + RC),$$
$$S(\lambda_{i,3},\mu_{i,3}) = 0,$$

for $i = 1, 2, \ldots, r$.

There are $16r$ linear equations and $16r$ quadratic equations on the finite field of $\mathbb{F}_q$ in $16(2r + 1)$ variables of λ_i, $i = 1, 2, \ldots, r$ and μ_i, $i = 0, 1, \ldots, r$. Let $\lambda_{i,j} = (\lambda_{i,j,0}, \lambda_{i,j,1}, \lambda_{i,j,2}, \lambda_{i,j,3})$ and $\mu_{i,j} = (\mu_{i,j,0}, \mu_{i,j,1}, \mu_{i,j,2}, \mu_{i,j,3})$. There are $16r$ quadratic monomials of $\lambda_{i,j,l} \cdot \mu_{i,j,l}$, where $\lambda_{i,j,l}, \mu_{i,j,l} \in \mathbb{F}_q$ for $i = 1, 2, \ldots, r, j = 0, 1, 2, 3, l = 0, 1, 2, 3$.

The round key for round i is

$$\sum_{j=0}^{i}\mu_j + RC = (\sum_{j=0}^{i}\mu_{j,0} + RC, \sum_{j=0}^{i}\mu_{j,1} + RC, \sum_{j=0}^{i}\mu_{j,2} + RC, \sum_{j=0}^{i}\mu_{j,3} + RC).$$

For each plaintext and ciphertext pair (P, C), let the input and output of the S-box layer for round i be $\beta_i, \alpha_i \in \mathbb{F}_q^{16}, i = 1, 2, \ldots, r$ (see Fig. 5), we have

$$\beta_1 = P + \mu_0,$$
$$SL(\beta_i, \alpha_i) = 0, i = 1, \ldots, r$$
$$\beta_{i+1} = LP(\alpha_i) + \sum_{j=0}^{i}\mu_j + RC, i = 1, \ldots, r-1$$
$$\alpha_r + \sum_{j=0}^{r}\mu_j + RC = C.$$

For each plaintext and ciphertext pair (P, C), we have the following $16r$ quadratic equations on the finite field,

$$SL(P + RK_0, \alpha_1) = 0,$$
$$SL(LP(\alpha_i) + RK_i, \alpha_{i+1}) = 0, i = 1, \ldots, r-2$$
$$SL(LP(\alpha_{r-1}) + RK_{r-1}, RK_r \oplus C) = 0.$$

There are $16(r + 1)$ linear equations and $16r$ quadratic equations on the finite field of $\mathbb{F}_q$ in $32r$ more variables of $\beta_i, \alpha_i, i = 1, 2, \ldots, r$. Let $\beta_{i,j} = (\beta_{i,j,0}, \beta_{i,j,1}, \beta_{i,j,2}, \beta_{i,j,3})$ and $\alpha_{i,j} = (\alpha_{i,j,0}, \alpha_{i,j,1}, \alpha_{i,j,2}, \alpha_{i,j,3})$. There are $16r$ quadratic monomials of $\beta_{i,j,l} \cdot \alpha_{i,j,l}$, where $\beta_{i,j,l}, \alpha_{i,j,l} \in \mathbb{F}_q$ for $i = 1, 2, \ldots, r, j = 0, 1, 2, 3, l = 0, 1, 2, 3$. For m pairs of (P, C), there are $16(r + 1)m$ linear equations and $16rm$ quadratic equations on the finite field of $\mathbb{F}_q$ in $32rm$ more variables and $16rm$ quadratic monomials.

Thus, there are in total $16(r + 1)m + 16r$ linear equations and $16rm + 16r$ quadratic equations on the finite field of $\mathbb{F}_q$ in $16(2r + 1) + 32rm$ variables and $16rm + 16r$ quadratic monomials. Since the number of equations $32rm+16m+32r$ is always less than the number of variables $48rm + 48r + 16$ for $r, m \geq 1$, thus our cipher is resistant to algebraic attack by linearization.

VII. Application and Implementation

We implemented the YuX using the open-source fully homomorphic encryption library HElib [36], which implemented the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [3]. Our implementation was tested on a Ubuntu Server with 256GB RAM (Architecture: 104 Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz).[4]

A. Benchmark

We present our homomorphic implementation of YuX below. The BGV scheme supports SIMD-style operations, which enables us to evaluate multiple instances in parallel. Our primary state-sliced implementation employs sixteen distinct HE ciphertexts (16 Ctxt) to represent a block state. We also experimented with two other "packed" implementations, which pack a block into one ciphertext (1 Ctxt), and "4 Ctxt", which packs a block into four ciphertexts. However, our results demonstrate that using sixteen distinct HE ciphertexts is the most efficient of the three implementations. As each ciphertext has *nslots* slots, these sixteen ciphertexts can hold the state in *nslots* different YuX blocks.

1) Parameters: For our implementation, we choose the "L" for the homomorphic encryption scheme, which is at least the multiplication depth of YuX. In addition, we choose the optimum parameter m to perform homomorphic encryption initialization so as to support operations on the finite field $\mathbb{F}_q$ and satisfy the following constraints.

- p is characteristic of plaintext space, so it equals 2 for $\mathsf{Yu_2X}$ or p for $\mathsf{Yu_pX}$.
- N is the degree of the polynomial modulus in the BGV scheme, which is equal to $\phi(m)$.
- d is the embedding degree. (Specially, For $\mathsf{Yu_2X}$-n, d should satisfy $n|d$. Since the plaintext state is operated over the finite field $\mathbb{F}_{2^n}$, the $\mathbb{F}_{2^n}$ must be a sub-field of $\mathbb{F}_{2^d}$.)

[4]Our implementation is available at https://github.com/YuXenc/Transcipher-Yux, and any future revisions or updates will be consistently released on this platform.

- L is the number of "levels", which means the number of levels in the modulus chain, and it is at least the multiplication depth of YuX.
- $\lceil \log q \rceil$ is the bit size of the primes in the modulus chain.
- λ' is the security parameter providing at least λ' bit security, which should be bigger than 128.
- $nslots$ is the number of plaintext slots. $nslots = \phi(m)/d$.

2) Implementation Details of YuX Decryption: For the homomorphic decryption of YuX, we utilize the 16 Ctxt version where each ciphertext holds s slots to represent a block state, as shown in Fig. 6. This approach enables the parallel evaluation of multiple blocks, thereby enhancing the efficiency of the homomorphic decryption process. Further details regarding this implementation are outlined below.

- Adding round-key is simply an addition of the round key and the HE ciphertexts that are the encrypted state.
- S-box layer has four parallel S-box S^{-1} composed of Pf iterating four times (i.e., $S^{-1} = Pf^4$) over $\mathbb{F}_q^4$. The multiplication depth of Pf^2 is 1, then the multiplication depth of S^{-1} is 2 (i.e., S^{-1} consumes 2 levels). The only nonlinear operation in S^{-1} is the multiplication over $\mathbb{F}_q$, and the number of multiplication in S^{-1} is 4. The decryption S-box is implemented by consuming 2 levels listed in Algorithm 4.
- Linear layer is implemented as Algorithm 5. The permutations in the linear layer step are "for free", which just has 96 homomorphic addictions and consumes 0 levels in the modulus chain.

Algorithm 4 Decryption S-Box

Input: $x = (x_0, x_1, x_2, x_3)$, round constant α
Output: $S^{-1}(x) = Pf^4(x)$
1: **for** $i = 1, \ldots, 4$ **do**
2: $tmp \leftarrow x_1 \cdot x_2$ // multiplication (-1 Level)
3: $tmp \leftarrow tmp + x_0 + x_3 + \alpha$ // addition (-0 Level)
4: $(x_0, x_1, x_2, x_3) \leftarrow (x_1, x_2, x_3, tmp)$
5: **end for**
6: **return** (x_0, x_1, x_2, x_3)

Algorithm 5 Decryption Linear Layer

Input: $x = (x_0, x_1, \ldots, x_{15})$
Output: $LP_{inv}(x)$
1: **for** $i = 0, 1, \ldots, 15$ **do**
2: $c_i \leftarrow \sum_{t \in \{0,3,4,8,9,12,14\}} x_{(i+t) mod 16}$,
3: **end for**
4: **return** $c = (c_0, \ldots, c_{15})$

A YuX decryption round function is implemented via a depth-2 circuit. The decryption of YuX consumes 24 levels for Yu_2X and 16 levels for Yu_pX in the modulus chain (2 levels per round), and all of them are consumed by the decryption S-box layer.

To reduce the ciphertext extension of round keys under FHE and the communication overhead, we pack all round keys into different slots of one FHE ciphertext orderly. Then in the server, it would be rearranged into different FHE ciphertexts to be used in later steps conveniently.

3) State-Sliced Packed Implementations: In the four ciphertexts implementation (4 Ctxt), we packed a block into four HE ciphertexts, which means $nslots/4$ different YuX blocks can be evaluated in parallel as

$$Ctxt_0=[x_0, x_4, x_8, x_{12}] \qquad Ctxt_1 = [x_1, x_5, x_9, x_{13}]$$
$$Ctxt_2=[x_2, x_6, x_{10}, x_{14}] \qquad Ctxt_3 = [x_3, x_7, x_{11}, x_{15}].$$

The S-box steps are exactly as above (except applied to 4 Ctxt), but the linear layer steps need extra slot rotation operations. In our tests, the throughput of 16 ciphertexts is about 8 times higher than the four ciphertexts version.

In one ciphertext implementation (1 Ctxt), we placed several YuX states in plaintext slots as

$$Ctxt = [\underbrace{x_0^0, x_1^0, \ldots, x_{15}^0}_{block^0}, \underbrace{x_0^1, x_1^1, \ldots, x_{15}^1}_{block^1}, \underbrace{x_0^2, x_1^2, \ldots, x_{15}^2}_{block^2}, \ldots].$$

The S-box layer and linear layer need extra slot rotation and Frobenius operations, and the number of packed blocks is reduced to at most 1/16 $nslots$. So the running time per bit is uncompetitive, and the throughput is much lower than the above two versions.

4) Implementing YuX Encryption: We implemented the YuX encryption circuit with the HElib library, and the HE encryption time was about 60% slower than HE decryption. The reason is that the S-box layer in YuX encryption has 2 more multiplications and Frobenius operations than the S-box layer in YuX decryption.

B. Comparison

To compare various symmetric cipher implementations concerning their suitability for FHE, we present a summary of the performance of selected ciphers that are FHE-friendly. As discussed in Sect. V-A, the efficiency of FHE implementations is mainly affected by the multiplication complexity and the running time of basic operations. Table VIII (which lists only the operations of block ciphers) shows that the expensive operation in YuX is only multiplication, and the total number of multiplications is competitive. We present a comprehensive summary of performance metrics in Table VI, which have been acquired through uniform execution environments. Specifically, the implementations for LowMC and AES were sourced from the public repositories maintained by their respective designers.[5] The reported figures for FLIP are approximations grounded in the work of [13]. In the case of Kreyvium, Rasta, Masta, Pasta, and HERA, the performance metrics are based on executions of implementations available within an open-source framework.[6] For CHAGHRI, its total time metric has been proportionally scaled based on a comparative analysis with AES, a choice made by its developers. It is important to note that Dasta has not been included in this comparison due to

[5]The implementations of LowMC and AES can be accessed at https://bitbucket.org/malb/lowmc-helib and https://github.com/homenc/HElib/tree/aes.

[6]Implementations can be found at https://github.com/IAIK/hybrid-HE-framework.git.

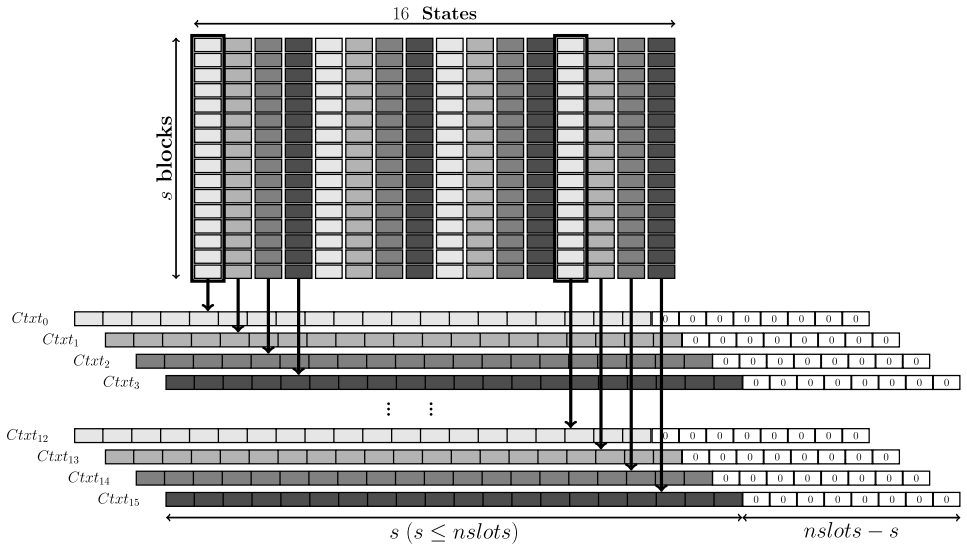


Fig. 6. The packing of plaintext states of YuX.

TABLE V
HOMOMORPHIC OPERATIONS AND MULTIPLICATION-DEPTH OF BLOCK CIPHERS

Ciphers(Parameters)	pt-ct Mul	ct-ct Mul	Rotation	Frobenius	Depth
AES(128,128,10)	210	40	60	110	40
LowMCv3(128,80,13)	-	1209	-	-	13
LowMCv3(256,128,14)	-	2646	-	-	14
CHAGHRI(189,189,8)	64	16	32	32	32
Yu_2X(128,128,12)	-	**192**	-	-	**24**
Yu_2X(256,128,12)	-	**192**	-	-	**24**
Yu_pX(256,118, 9)	-	**144**	-	-	**18**

its primary focus on enhancing client-side performance; consequently, it does not surpass the performance of Rasta on the server side. In terms of Fully Homomorphic Encryption (FHE) evaluation throughput, Yu_2X-8 and Yu_2X-16 achieve rates of 12.80 KB/min and 12.33 KB/min, which are approximately 12, 17, and 9 times higher than that of AES-128, CHAGHRI, and LowMC-128 (with 128-bit security), respectively. Furthermore, the throughput of Yu_pX achieves 858 KB/min. Notably, YuX demonstrates superior performance achievements.

The notable throughput of YuX can be attributed to a combination of factors. Firstly, the word-orientation of YuX enables its homomorphic implementation with elements from the field $\mathbb{F}_q$ in each slot of the ciphertext. The employment of a state-sliced implementation proves to be more efficient than the packed alternative. In contrast, for the Masta, Pasta, and HERA ciphers (operating over $\mathbb{F}_p^t$), the packed implementation entails an evaluation of the non-linear layer that requires $O(1)$ FHE operations, whereas the state-sliced implementation demands $O(t)$ FHE operations. Of greater significance is the fact that our linear layer of decryption is adeptly engineered to function without necessitating any multiplications. This latter facet is instrumental in augmenting the performance and computational efficiency of the system.

C. Applications

1) Data Integrity Verification: In outsourcing scenarios, such as cloud storage, data integrity verification is of paramount importance [48]. In these environments, user data is outsourced to third-party service providers, such as cloud service providers. These service providers may obtain data from the user and store it in their facilities. However, for a variety of reasons including, but not limited to, errors, malicious operations, or attacks, data stored in the cloud may be tampered with or deleted. The transciphering framework could be used for data integrity verification in outsourcing scenarios.

To verify data integrity, we can use Message Authentication Codes (MACs) such as GHash [49]. Before being stored, data is encrypted using a block cipher (such as YuX) and its MAC is computed, with the MAC serving as a fingerprint or signature of the data. When it is necessary to verify the data's integrity, the MAC of the encrypted data stored in the cloud can be recomputed and compared with the original MAC. If the two match, the data is considered intact. The core operation of computing MAC on the ciphertext is implemented using Fully Homomorphic Encryption (FHE). This process necessitates the transformation of YuX ciphertexts into FHE ciphertexts, a process achievable by using the transciphering framework.

TABLE VI
COMPARISON OF VARIOUS SYMMETRIC CIPHERS BASED ON BGV

Ciphers	(Parameters)	p	N	$\lceil \log q \rceil$	$nslots$(Blocks)	λ'	$t_{eval}(s)$	Throughput (KB/min)
AES	(128,128,10)	2	46080	879	1920(120)	151	106.73	1.05*
LowMCv3	(128,80,13)	2	18000	406	720	112	335.02	2.01
LowMCv3	(256,80,13)	2	18000	406	720	112	634.20	2.12
LowMCv3	(256,128,14)	2	21168	437	756	132	997.74	1.42
Kreyvium†	(42,128,12)	2	14112	250	504	108	478.43	0.32
Kreyvium†	(125,128,13)	2	15004	270	682	113	566.36	1.10
Kreyvium†	(406,128,6)	2	-	-	720	-	1263.26	1.69
FLIP†	(1,128,4)	2	-	-	630	-	3.03	1.52
Rasta†	(525,128,5)	2	14112	200	504	139	351.14	5.51
Rasta†	(351,128,6)	2	18000	270	720	140	442.73	4.18
CHAGHRI	(189,189,8)	2	47628	700	126(42)	174	78.24	0.743*
Yu_2X-8	**(128,128,12)**	2	**23040**	**500**	**960**	**139**	**70.33**	**12.80**
Yu_2X-16	**(256,128,12)**	2	**24576**	**500**	**512**	**135**	**77.86**	**12.33**
Yu_2X-16	**(256,128,12)**	2	**32768**	**520**	**2048**	**199**	**143.40**	**26.78**
Yu_2X-16	**(256,128,14)**	2	**32768**	**620**	**2048**	**150**	**203.14**	**18.90**
Masta†	(16*128,128,4)	65537	32768	470	32768(256)	163	33.38	115.04*
Masta†	(16*64,128,5)	65537	32768	550	32768(512)	133	28.50	134.74*
Masta†	(16*32,128,6)	65537	32768	450	32768	169	773.83	9.92
Masta†	(16*16,128,7)	65537	32768	500	32768	144	242.6	15.82
Pasta†	(16*16, 128,3)	65537	32768	400	32768(2048)	184	25.66	149.65*
HERA†	(16*16,128,5)	65537	65536	600	32768(2048)	254	20.23	189.82*
Yu_pX-65537	**(16*16,118,9)**	**65537**	**32768**	**853**	**32768**	**118**	**71.57**	**858.46**
Yu_pX-65537	**(16*16,128,12)**	**65537**	**65536**	**1120**	**32768**	**164**	**221.65**	**277.19**
Yu_pX-65537	**(16*16,128,14)**	**65537**	**65536**	**1300**	**32768**	**142**	**308.50**	**199.16**

Cipher (n, d, r): n, d, and r denote the block size, data complexity, and round number;
(*Blocks*): the number of blocks computed in parallel;
t_{eval}: the total running time of FHE-decryption circuit;
Throughput: Throughput= $nslots * \frac{n*2^{-13}*60}{t_{eval}}$. The implementations of the cipher are state-sliced or bit-sliced.
*Throughput**: Throughput= Blocks$*\frac{n*2^{-13}*60}{t_{eval}}$. The implementations of the cipher are packed.
Cipher†: Stream cipher.

GHash is a special kind of MAC that operates over a Galois field, such as $\mathbb{F}_2^8$ [50], enabling efficient integrity verification on encrypted data. By utilizing the transciphering framework, Yu_2X-8 encrypted data is transformed into fully homomorphically encrypted data using the implementation in VII-A, after which homomorphic GHash computations can be performed. Given that the GHash expression is

$$GHASH_H(X_1||X_2||X_3||\dots||X_m) = Y_m,$$

and the function is calculated as

$$Y_m = X_1 \cdot H^m \oplus X_2 \cdot H^{m-1} \oplus \cdots \oplus X_m \cdot H.$$

The FHE H^i operation can be substantially depth-reduced by utilizing homomorphic automorphism, such as the depth of $H^{33} = H^{2^5} \cdot H$ is 2. The implementation of homomorphic GHash can be realized by using homomorphic automorphism and multiplication operations, yielding a homomorphically encrypted MAC for the user to verify data integrity.

2) Private Set Intersection: The transciphering framework would be convenient for some applications that operate over finite fields. Especially for MPC applications [51], [52], [53], [54] (like SPDZ [55], [56]) and FHE applications [57], [58], [59], extension fields of characteristic two and a prime number are frequent to represent information-theoretic tags, e.g., hashing strings and secrets. We now introduce a cryptographic protocol called private set intersection (PSI) [57], [58], [59], [60] and show how to integrate with our transciphering framework. Private set intersection refers to a functionality that the client and the server, holding their own set X and Y respectively, retrieve the intersection of two sets without revealing anything else to each other.

One popular solution [57] is that the client encrypts each entry (i.e., consumes one or multiple slots) by employing BGV/FV scheme, and sends the ciphertext to the server. The server, holding the set $Y = \{y_i\}_{i\in[n]}$, now homomorphically evaluate the polynomial function $T(A) = r\prod_{i\in[n]}(A - y_i)$, where r is a non-zero random element. Actually, we can easily know $T(A) = 0$ if $A \in Y$, and $T(A)$ is a non-zero random element, otherwise. The client can obtain the intersection $X \cap Y$ by checking if each decrypted slot is equal to 0. Further, some optimizations can be exploited, but we only introduce high-level ideas and don't stuck in too many details,

TABLE VII
THE PERFORMANCE OF PRIVATE SET INTERSECTION PROTOCOL WITH TRANSCIPHERING YuX

$\|Y\|$	$\|X\|$	Table size	Insert size	Communication (KB)		Server online (s)		False positive
				query	response	transciphering	PSI	
2^{20}	11041	16384	318	256	4352	815	1452	2^{-61}
	5535	8192	556	128	2176	412	1385	2^{-55}
2^{16}	11041	16384	51	256	4352	822	414	2^{-82}
	5535	8192	74	128	2176	411	266	2^{-78}

Insert size stands for the size in each table when exploit cuckoo hashing.

which is enough to understand how to integrate with our transciphering algorithm YuX. (1)Preprocessing. The server rewrites the polynomial function $T(A) = rA^n + \sum_{i\in[n]} rb_iA^i$, where the coefficients $\{b_i\}_{i\in[n]}$ are only determined by the set Y, so they are usually preprocessed by the server. (2)Cuckoo hashing [57], [61]. Cuckoo hashing is widely used to greatly reduce the amortized computational complexity of each query when the client wants to ask multiple queries at once. For example, supposing that $|X| = 11041$ and $|Y| = 2^{20}$, each element in X now only need to intersect with around 318 items [57] instead of 2^{20} items, with a hashing failure probability 2^{-40}. (3)Frobenius automorphism and Paterson-Stockmeyer algorithm [59], [60]. The two algorithms contribute to lower multiplication depth and lower computation cost. (4)We don't exploit windowing, partitioning and extremal postage-stamp bases methods [59], for these methods trade higher communication for smaller computation. We don't implement labeled mode, nor apply OPRF to resist malicious clients, but integration with the two extensions is trivial and we leave them to further work.

Our transciphering algorithm can be directly integrated with the PSI protocol mentioned above. The client sends the ciphertexts from the symmetric cipher instead of RLWE ciphertexts, and the server performs our transciphering algorithm in front of the PSI protocol. We focus on the communication and server online time when or not employing our transciphering algorithm. Supposing that each item in two sets is a 128-bit hashing string, we break each hashing string to multiple 16-bit strings and encode them to different slots. We implement a proof-of-concept variant of PSI protocol with transciphering algorithm $\mathsf{Yu_2X}$-16 upon HElib library, and the concrete results are shown in Table VII. Our communication size is much smaller, e.g., our communication is 4.5 MB when the client want to intersect 11041 items to 2^{20} itmes, which is smaller than 11.5 MB [57], [60]. Moreover, each item in our sets is 128-bit string, which is larger than 32 bit [57]. The disadvantage of our protocol is that our server online time is slower. In order to achieve better performance, one direction is to implement our protocol by employing SEAL library [62] with HEXL acceleration [63], the another is to use GPU [64] and hardware accelerator [65] in the further.

It is worth noting that when breaking a hashing string into multiple 16-bit segments and encoding them into separate slots, there exists a probability of false positive [60], meaning it may return a false positive response indicating an item exists in the server's set when, in fact, it does not. The tolerance of false positive depends on the application scenario, but 16 bits is sufficiently large that false positive is easily smaller than 2^{-40}. Concrete probability of false positive can be found in Table VII.

VIII. CONCLUSION

This paper introduces YuX, a block cipher based on the traditional SPN structure and defined over $\mathbb{F}_q^{16}$, which optimizes the decryption circuit's FHE evaluation, and then we give two instances named $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$ where q is 2^n and a prime p, respectively. Our detailed implementations demonstrate that $\mathsf{Yu_2X}$ and $\mathsf{Yu_pX}$ are currently the fastest block cipher for FHE evaluation of the decryption circuit over characteristic-2 ciphers and characteristic-p ciphers. The potential applications and further evaluations of YuX in MPC and ZK warrant additional investigation. Moreover, the design strategy presented in this paper can be utilized to develop additional symmetric primitives with lower multiplication depth and complexity.

APPENDIX A
THE ANFS OF S-BOX INVERSE OF $\mathsf{Yu_2X}$-8

The S-box inverse S^{-1} is constructed by iterating the Pf 4 times, we have that

$$\mathrm{Pf}(x_0, x_1, x_2, x_3) = (x_1, x_2, x_3, x_0 + x_1 \odot x_2 + x_3 + \alpha)$$

where $\alpha = x^7 + x^6 + x^3 + x^2 + 1 \in \mathbb{F}_{2^8}$ is a constant. In Pf, $\odot$ is the multiplication over $\mathbb{F}_{2^8}$ and the defining polynomial of $\mathbb{F}_{2^8}$ is $x^8 + x^4 + x^3 + x + 1$. Denote

$$\begin{cases} a &= (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7), \\ b &= (b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7), \\ c &= (c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7). \end{cases}$$

The ANF of $c = a \odot b$ is shown by:

$$\begin{aligned}
c_0 &= a_0b_0 + a_1b_7 + a_2b_6 + a_3b_5 + a_4b_4 + a_5b_3 + a_5b_7 \\
&\quad + a_6b_2 + a_6b_6 + a_6b_7 + a_7b_1 + a_7b_5 + a_7b_6, \\
c_1 &= a_0b_1 + a_1b_0 + a_1b_7 + a_2b_6 + a_2b_7 + a_3b_5 + a_3b_6 \\
&\quad + a_4b_4 + a_4b_5 + a_5b_3 + a_5b_4 + a_5b_7 + a_6b_2 + a_6b_3 \\
&\quad + a_6b_6 + a_7b_1 + a_7b_2 + a_7b_5 + a_7b_7, \\
c_2 &= a_0b_2 + a_1b_1 + a_2b_0 + a_2b_7 + a_3b_6 + a_3b_7 + a_4b_5 \\
&\quad + a_4b_6 + a_5b_4 + a_5b_5 + a_6b_3 + a_6b_4 + a_6b_7 + a_7b_2 \\
&\quad + a_7b_3 + a_7b_6, \\
c_3 &= a_0b_3 + a_1b_2 + a_1b_7 + a_2b_1 + a_2b_6 + a_3b_0 + a_3b_5
\end{aligned}$$

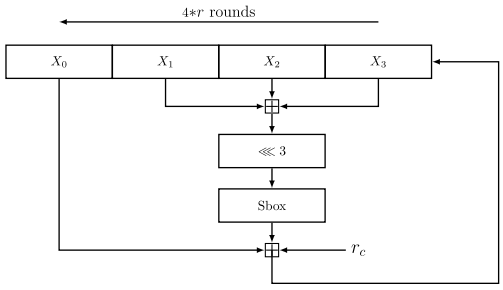


Fig. 7. Key schedule of YuX.

$$
\begin{aligned}
&+a_3b_7 + a_4b_4 + a_4b_6 + a_4b_7 + a_5b_3 + a_5b_5 + a_5b_6\\
&+a_5b_7 + a_6b_2 + a_6b_4 + a_6b_5 + a_6b_6 + a_6b_7 + a_7b_1\\
&+a_7b_3 + a_7b_4 + a_7b_5 + a_7b_6 + a_7b_7,\\
c_4 = {}& a_0b_4 + a_1b_3 + a_1b_7 + a_2b_2 + a_2b_6 + a_2b_7 + a_3b_1\\
&+a_3b_5 + a_3b_6 + a_4b_0 + a_4b_4 + a_4b_5 + a_4b_7 + a_5b_3\\
&+a_5b_4 + a_5b_6 + a_6b_2 + a_6b_3 + a_6b_5 + a_7b_1 + a_7b_2\\
&+a_7b_4 + a_7b_7,\\
c_5 = {}& a_0b_5 + a_1b_4 + a_2b_3 + a_2b_7 + a_3b_2 + a_3b_6 + a_3b_7\\
&+a_4b_1 + a_4b_5 + a_4b_6 + a_5b_0 + a_5b_4 + a_5b_5 + a_5b_7\\
&+a_6b_3 + a_6b_4 + a_6b_6 + a_7b_2 + a_7b_3 + a_7b_5,\\
c_6 = {}& a_0b_6 + a_1b_5 + a_2b_4 + a_3b_3 + a_3b_7 + a_4b_2 + a_4b_6\\
&+a_4b_7 + a_5b_1 + a_5b_5 + a_5b_6 + a_6b_0 + a_6b_4 + a_6b_5\\
&+a_6b_7 + a_7b_3 + a_7b_4 + a_7b_6,\\
c_7 = {}& a_0b_7 + a_1b_6 + a_2b_5 + a_3b_4 + a_4b_3 + a_4b_7 + a_5b_2\\
&+a_5b_6 + a_5b_7 + a_6b_1 + a_6b_5 + a_6b_6 + a_7b_0 + a_7b_4\\
&+a_7b_5 + a_7b_7.
\end{aligned}
$$

APPENDIX B
THE KEY SCHEDULE OF YuX

See Fig. 7.

APPENDIX C
THE MATRIX M^{-1} OF YuPX

As shown in the equation at the top of the next page.

APPENDIX D
COMPARISON OF STATE-SLICED AND PACKED

In VII-B, we present two computational formulas for throughput at Table VI, denoted as

$$Throughput = nslots * \frac{n * 2^{-13} * 60}{t_{eval}} \tag{5}$$

and

$$Throughput = s' * \frac{n * 2^{-13} * 60}{t_{eval}}. \tag{6}$$

These formulas are provided based on two distinct implementation approaches. Formula 5 is suited for the state-sliced implementation which is the method employed in the implementation of the LowMC and Yux. Formula 6, on the other hand, is suited for the packed implementation which is the method used in the implementation of Pasta and HERA. Taking the HERA algorithm as an example, we provide a detailed comparison of these two implementation approaches and their respective impact on throughput.

TABLE VIII
HOMOMORPHIC OPERATIONS IN S-BOX LAYER OF BLOCK CIPHERS

Ciphers	Implement	ct-ct Mul	Rotation
HERA	Padcked	2	-
	State-sliced	**32**	-
Pasta ($r-1$ round)	Packed	1	1
	State-sliced	**30**	-
Pasta (last round)	Padcked	2	-
	State-sliced	**2*32**	-
YuPX	State-sliced	18	-

A. Packed Implementation of S-Box Layer (HERA)

In packed implementation, it place several blocks in just one ciphertext as

$$Ctxt = [\underbrace{x_0^0, x_1^0, \ldots, x_{15}^0}_{block_0}, \underbrace{x_0^1, x_1^1, \ldots, x_{15}^1}_{block_1}, \ldots, \underbrace{x_0^{s'}, x_1^{s'}, \ldots, x_{15}^{s'}}_{block_{s'}}].$$

The homomorphic S-box implementation only need 2 homomorphic multiplicative using packed implementation, as

$$
\begin{aligned}
&Ctxt * Ctxt * Ctxt\\
&= [\underbrace{(x_0^0)^3, (x_1^0)^3, \ldots, (x_{15}^0)^3}_{block_0}, \underbrace{(x_0^1)^3, (x_1^1)^3, \ldots, (x_{15}^1)^3}_{block_1},\\
&\ldots, \underbrace{(x_0^{s'})^3, (x_1^{s'})^3, \ldots, (x_{15}^{s'})^3}_{block_{s'}}].
\end{aligned}
$$

B. State-Sliced Implementation of S-Box Layer (HERA)

In the state-sliced implementation, it place several blocks in 16 ciphertexts as

$$
\begin{cases}
Ctxt_0 = [x_0^0, x_0^1, x_0^2, \ldots, x_0^{nslots-1}]\\
Ctxt_1 = [x_1^0, x_1^1, x_1^2, \ldots, x_1^{nslots-1}]\\
\ldots\ldots\\
Ctxt_{14} = [x_{14}^0, x_{14}^1, x_{14}^2, \ldots, x_{14}^{nslots-1}]\\
Ctxt_{15} = [x_{15}^0, x_{15}^1, x_{15}^2, \ldots, x_{15}^{nslots-1}].
\end{cases}
$$

Since each ciphertext can hold $nslots$ plaintext slots, these 16 ciphertexts can hold the state of $nslots$ different blocks. The homomorphic S-box implementation need 2*16 homomorphic

$$M^{-1} = \begin{bmatrix}
\frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} \\
\frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} \\
\frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} \\
\frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} \\
-\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} \\
-\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} \\
-\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} \\
-\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} \\
-\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} \\
\frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} \\
\frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} \\
\frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} \\
\frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} \\
\frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} \\
\frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} \\
\frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & -\frac{3}{7} & -\frac{16}{21} & -\frac{16}{21} & -\frac{16}{21} & -\frac{3}{7} & \frac{5}{21} & \frac{5}{21} & \frac{5}{21} & \frac{4}{7}
\end{bmatrix}$$

multiplicative using state-sliced implementation, as

$$\begin{cases}
Ctxt_0 * Ctxt_0 * Ctxt_0 \\
\quad = [(x_0^0)^3, (x_0^1)^3, (x_0^2)^3, \ \ldots, (x_0^{nslots-1})^3] \\
Ctxt_1 * Ctxt_1 * Ctxt_1 \\
\quad = [(x_1^0)^3, (x_1^1)^3, (x_1^2)^3, \ \ldots, (x_1^{nslots-1})^3] \\
\ldots \qquad \ldots \\
Ctxt_{14} * Ctxt_{14} * Ctxt_{14} \\
\quad = [(x_{14}^0)^3, (x_{14}^1)^3, (x_{14}^2)^3, \ \ldots, (x_{14}^{nslots-1})^3] \\
Ctxt_{15} * Ctxt_{15} * Ctxt_{15} \\
\quad = [(x_{15}^0)^3, (x_{15}^1)^3, (x_{15}^2)^3, \ \ldots, (x_{15}^{nslots-1})^3]
\end{cases}$$

Using state-sliced implementation would result in a substantial increment in the number of homomorphic multiplicative operations in the S-box layer, specifically by a factor of 16.

The efficiency of FHE implementations is primarily influenced by the complexity of multiplication operations. Table VIII illustrates the homomorphic operations of S-box layer involved in two implementations of HERA, Pasta, and YuX. The comparison reveals a significant increase in homomorphic ciphertext multiplications (ct-ct Mul) within the State-sliced implementations of HERA and Pasta when contrasted with their Packed implementations.

REFERENCES

[1] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 9056, E. Oswald and M. Fischlin, Eds. Berlin, Germany: Springer, Apr. 2015, pp. 617–640.

[2] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, Jan. 2020.

[3] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, S. Goldwasser, Ed. New York, NY, USA: ACM, Jan. 2012, pp. 309–325.

[4] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 7417, R. Safavi-Naini and R. Canetti, Eds. Santa Barbara, CA, USA: Springer, Aug. 2012, pp. 868–886.

[5] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptol. ePrint Arch., Tech. Rep. 2012/144, 2012. [Online]. Available: https://eprint.iacr.org/2012/144

[6] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 10624, T. Takagi and T. Peyrin, Eds. Cham, Switzerland: Springer, Dec. 2017, pp. 409–437.

[7] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proc. 3rd ACM workshop Cloud Comput. Secur. Workshop*. New York, NY, USA: Association for Computing Machinery, Oct. 2011, pp. 113–124, doi: 10.1145/2046660.2046682.

[8] J. Cho et al., "Transciphering framework for approximate homomorphic encryption," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 13092, M. Tibouchi and H. Wang, Eds. Cham, Switzerland: Springer, Dec. 2021, pp. 640–669.

[9] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 7417, R. Safavi-Naini and R. Canetti, Eds. Berlin, Germany: Springer, Aug. 2012, pp. 850–867.

[10] Y. Doröz, A. Shahverdi, T. Eisenbarth, and B. Sunar, "Toward practical homomorphic evaluation of block ciphers using prince," in *Proc. Int. Conf. Financial Cryptogr. Data Secur. Workshops*, in Lecture Notes in Computer Science, vol. 8438, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds. Berlin, Germany: Springer, Heidelberg, Mar. 2014, pp. 208–220.

[11] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner, "Ciphers for MPC and FHE," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 9056, E. Oswald and M. Fischlin, Eds. Berlin, Germany: Springer, Apr. 2015, pp. 430–454.

[12] A. Canteaut et al., "Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 9783, T. Peyrin, Ed. Berlin, Germany: Springer, Mar. 2016, pp. 313–333.

[13] P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet, "Towards stream ciphers for efficient FHE with low-noise ciphertexts," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 9665, M. Fischlin and J.-S. Coron, Eds. Berlin, Germany: Springer, May 2016, pp. 311–343.

[14] C. Dobraunig et al., "Rasta: A cipher with low ANDdepth and few ANDs per bit," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 10991, H. Shacham and A. Boldyreva, Eds. Cham, Switzerland: Springer, Aug. 2018, pp. 662–692.

[15] P. Hebborn and G. Leander, "Dasta—Alternative linear layer for Rasta," *IACR Trans. Symmetric Cryptol.*, vol. 2020, no. 3, pp. 46–86, Sep. 2020.

[16] C. Cid, J. P. Indrøy, and H. Raddum, "FASTA—A stream cipher for fast FHE evaluation," Cryptol. ePrint Arch., Tech. Rep. 2021/1205, 2021. [Online]. Available: https://eprint.iacr.org/2021/1205

[17] I. Dinur, Y. Liu, W. Meier, and Q. Wang, "Optimized interpolation attacks on LowMC," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 9453, T. Iwata and J. H. Cheon, Eds. Berlin, Germany: Springer, Nov./Dec. 2015, pp. 535–560.

[18] C. Dobraunig, M. Eichlseder, and F. Mendel, "Higher-order cryptanalysis of LowMC," in *Information Security and Cryptology—ICISC* (Lecture Notes in Computer Science), vol. 9558, S. Kwon and A. Yun, Eds. Cham, Switzerland: Springer, Nov. 2016, pp. 87–101.

[19] C. Rechberger, H. Soleimany, and T. Tiessen, "Cryptanalysis of low-data instances of full LowMCv2," *IACR Trans. Symmetric Cryptol.*, vol. 2018, no. 3, pp. 163–181, Sep. 2018.

[20] M. Liu, "Degree evaluation of NFSR-based cryptosystems," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 10403, J. Katz and H. Shacham, Eds. Cham, Switzerland: Springer, Aug. 2017, pp. 227–249.

[21] Y. Todo, T. Isobe, Y. Hao, and W. Meier, "Cube attacks on non-blackbox polynomials based on division property," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 10403, J. Katz and H. Shacham, Eds. Cham, Switzerland: Springer, Aug. 2017, pp. 250–279.

[22] Y. Hao, Y. Todo, C. Li, T. Isobe, W. Meier, and Q. Wang, "Improved division property based cube attacks exploiting algebraic properties of superpoly," in *Proc. 38th Annu. Int. Cryptol. Conf.*, in Lecture Notes Computer Science, vol. 10991, Santa Barbara, CA, USA. Cham, Switzerland: Springer, Aug. 2018, pp. 275–305.

[23] S. Duval, V. Lallemand, and Y. Rotella, "Cryptanalysis of the FLIP family of stream ciphers," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 9814, M. Robshaw and J. Katz, Eds. Berlin, Germany: Springer, Aug. 2016, pp. 457–475.

[24] F. Liu, S. Sarkar, W. Meier, and T. Isobe, "Algebraic attacks on Rasta and Dasta using low-degree equations," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 13090, M. Tibouchi and H. Wang, Eds. Cham, Switzerland: Springer, Dec. 2021, pp. 214–240.

[25] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 10031, J. H. Cheon and T. Takagi, Eds. Berlin, Germany: Springer, Dec. 2016, pp. 191–219.

[26] M. R. Albrecht et al., "Feistel structures for MPC, and more," in *Computer Security—ESORICS* (Lecture Notes in Computer Science), vol. 11736, K. Sako, S. Schneider, and P. Y. A. Ryan, Eds. Cham, Switzerland: Springer, Sep. 2019, pp. 151–171.

[27] M. R. Albrecht et al., "Algebraic cryptanalysis of STARK-friendly designs: Application to MARVELLOUS and MiMC," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 11923, S. D. Galbraith and S. Moriai, Eds. Cham, Switzerland: Springer, Dec. 2019, pp. 371–397.

[28] T. Ashur, M. Mahzoun, and D. Toprakhisar, "Chaghri—An FHE-friendly block cipher," Cryptol. ePrint Arch., Tech. Rep. 2022/592, 2022. [Online]. Available: https://eprint.iacr.org/2022/592

[29] J. Ha et al., "Masta: An HE-friendly cipher using modular arithmetic," *IEEE Access*, vol. 8, pp. 194741–194751, 2020.

[30] C. Dobraunig, L. Grassi, L. Helminger, C. Rechberger, M. Schofnegger, and R. Walch, "Pasta: A case for hybrid homomorphic encryption," Cryptol. ePrint Arch., Tech. Rep. 2021/731, 2021. [Online]. Available: https://eprint.iacr.org/2021/731

[31] M. Eichlseder et al., "An algebraic attack on ciphers with low-degree round functions: Application to full MiMC," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 12491, S. Moriai and H. Wang, Eds. Cham, Switzerland: Springer, Dec. 2020, pp. 477–506.

[32] F. Liu, R. Anand, L. Wang, W. Meier, and T. Isobe, "Coefficient grouping: Breaking Chaghri and more," Cryptol. ePrint Arch., Tech. Rep. 2022/991, 2022. [Online]. Available: https://eprint.iacr.org/2022/991

[33] J. Ha, S. Kim, B. Lee, J. Lee, and M. Son, "Rubato: Noisy ciphers for approximate homomorphic encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 13275, O. Dunkelman and S. Dziembowski, Eds. Cham, Switzerland: Springer, May/Jun. 2022, pp. 581–610.

[34] L. Grassi, I. M. Ayala, M. N. Hovd, M. Øygarden, H. Raddum, and Q. Wang, "Cryptanalysis of symmetric primitives over rings and a key recovery attack on rubato," Cryptol. ePrint Arch., Paper 2023/822, 2023. [Online]. Available: https://eprint.iacr.org/2023/822

[35] N. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," Cryptol. ePrint Arch., Tech. Rep. 2011/133, 2011. [Online]. Available: https://eprint.iacr.org/2011/133

[36] S. Halevi and V. Shoup, "Design and implementation of HElib: A homomorphic encryption library," Cryptol. ePrint Arch., Tech. Rep. 2020/1481, 2020. [Online]. Available: https://eprint.iacr.org/2020/1481

[37] K. Nyberg, "Differentially uniform mappings for cryptography," in *Advances in Cryptology—EUROCRYPT*, T. Helleseth, Ed. Berlin, Germany: Springer, 1994, pp. 55–64.

[38] M. Matsui, "New structure of block ciphers with provable security against differential and linear cryptanalysis," in *Fast Software Encryption*, D. Gollmann, Ed. Berlin, Germany: Springer, 1996, pp. 205–218.

[39] T. Baignères, J. Stern, and S. Vaudenay, "Linear cryptanalysis of non binary ciphers," in *Selected Areas in Cryptography*, C. Adams, A. Miri, and M. Wiener, Eds. Berlin, Germany: Springer, 2007, pp. 184–211.

[40] C. Carlet, P. Charpin, and V. Zinoviev, "Codes, bent functions and permutations suitable for DES-like cryptosystems," *Des., Codes Cryptogr.*, vol. 15, no. 2, pp. 125–156, Nov. 1998.

[41] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *J. Cryptol.*, vol. 4, no. 1, pp. 3–72, Jan. 1991.

[42] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 765, T. Helleseth, Ed. Heidelberg, Germany: Springer, May 1994, pp. 386–397.

[43] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, vol. 1592. Prague, Czech Republic. Cham, Switzerland: Springer, May 1999, pp. 12–23, doi: 10.1007/3-540-48910-X_2.

[44] L. Knudsen, "DEAL—A 128-bit block cipher," *Complexity*, vol. 258, no. 2, 1998.

[45] A. Bogdanov and V. Rijmen, "Linear hulls with correlation zero and linear cryptanalysis of block ciphers," *Designs, Codes Cryptogr.*, vol. 70, no. 3, pp. 369–383, Mar. 2014.

[46] Y. Todo, "Structural evaluation by generalized integral property," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 9056, E. Oswald and M. Fischlin, Eds., Berlin, Germany, Apr. 2015, pp. 287–314.

[47] Z. Xiang, W. Zhang, Z. Bao, and D. Lin, "Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers," in *Proc. 22nd Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, in Lecture Notes Computer Science, vol. 10031. Berlin, Germany: Springer, Dec. 2016, pp. 648–678.

[48] N. Garg, A. Nehra, M. Baza, and N. Kumar, "Secure and efficient data integrity verification scheme for cloud data storage," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2023, pp. 1–6.

[49] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (GCM) of operation," in *Proc. Int. Conf. Cryptol.*, Chennai, India. Berlin, Germany: Springer, 2004, pp. 343–355.

[50] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST SP 800-38D, 2007.

[51] M. Keller and A. Yanai, "Efficient maliciously secure multiparty computation for RAM," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 10822, J. B. Nielsen and V. Rijmen, Eds. Heidelberg, Germany: Springer, Apr./May 2018, pp. 91–124.

[52] D. Chaum, C. Crépeau, and I. Damgard, "Multiparty unconditionally secure protocols (extended abstract)," in *Proc. 20th ACM STOC*. New York, NY, USA: ACM Press, May 1988, pp. 11–19.

[53] K. Chida et al., "Fast large-scale honest-majority MPC for malicious adversaries," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 10993, H. Shacham and A. Boldyreva, Eds. Heidelberg, Germany: Springer, Aug. 2018, pp. 34–64.

[54] M. Abspoel, A. P. K. Dalskov, D. Escudero, and A. Nof, "An efficient passive-to-active compiler for honest-majority MPC over rings," in *Proc. ACNS*, in Lecture Notes in Computer Science, vol. 12727, K. Sako and N. O. Tippenhauer, Eds. Heidelberg, Germany: Springer, Jun. 2021, pp. 122–152.

[55] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. New York, NY, USA: ACM Press, Nov. 2020, pp. 1575–1590.

[56] (2020). *MP-SPDZ*. [Online]. Available: https://github.com/data61/MP-SPDZ

[57] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. New York, NY, USA: ACM Press, Oct. 2017, pp. 1243–1255.

[58] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled PSI from fully homomorphic encryption with malicious security," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. New York, NY, USA: ACM Press, Oct. 2018, pp. 1223–1237.

[59] K. Cong et al., "Labeled PSI from homomorphic encryption with reduced computation and communication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, G. Vigna and E. Shi, Eds. New York, NY, USA: ACM Press, Nov. 2021, pp. 1135–1150.

[60] (2021). *APSI*. [Online]. Available: https://github.com/microsoft/APSI

[61] R. Pagh and F. F. Rodler, "Cuckoo hashing," in *Proc. 9th Annu. Eur. Symp. Algorithms (ESA)*, in Lecture Notes in Computer Science, vol. 2161, F. M. auf der Heide, Ed., Aarhus, Denmark. Berlin, Germany: Springer, Aug. 2001, pp. 121–133, doi: 10.1007/3-540-44676-1_10.

[62] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library—SEAL v2.1," in *Proc. Int. Conf. Financial Cryptogr.*, in Lecture Notes in Computer Science, vol. 10323, M. Brenner et al., Eds. Heidelberg, Germany: Springer, Apr. 2017, pp. 3–18.

[63] F. Boemer, S. Kim, G. Seifu, F. D. de Souza, and V. Gopal, "Intel HEXL: Accelerating homomorphic encryption with Intel AVX512-IFMA52," Cryptol. ePrint Arch., Tech. Rep. 2021/420, 2021. [Online]. Available: https://eprint.iacr.org/2021/420

[64] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 4, pp. 114–148, Aug. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9062

[65] D. B. Cousins et al., "TREBUCHET: Fully homomorphic encryption accelerator for deep computation," Cryptol. ePrint Arch., Paper 2023/521, 2023. [Online]. Available: https://eprint.iacr.org/2023/521

Fen Liu received the B.S. and M.S. degrees from Xidian University, Xi'an, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree with the Institute of Information Engineering, Chinese Academy of Sciences. Her current research interests include the design and analysis of block ciphers.

Yongqiang Li received the B.S. and M.S. degrees in mathematics from Beijing Normal University, Beijing, China, in 2005 and 2008, respectively, and the Ph.D. degree in information security from the Institute of Software, Chinese Academy of Sciences, China, in January 2012. He is currently an Associate Professor with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include symmetric cryptography and related areas.

Huiqin Chen received the B.S. degree from Xidian University. She is currently pursuing the Ph.D. degree with the Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences. Her primary research interests include the design of components for block ciphers and the development of automated techniques for cryptographic analysis.

Lin Jiao received the B.S. degree from Jilin University, Jilin, China, in 2010, and the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in 2016. She is currently an Associate Professor with the State Key Laboratory of Cryptology, Beijing. Her research interests include symmetric cryptography and related areas.

Ming Luo received the B.S. degree from the University of Electronic Science and Technology of China in 2018. He is currently pursuing the Ph.D. degree with the Chinese Academy of Sciences. His research interests include fully homomorphic encryption and private information retrieval.

Mingsheng Wang received the Ph.D. degree from Beijing Normal University, Beijing, China, in 1994. He is currently a Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing. His research interests include computational algebra, cryptography, and information security.