Minimize the Randomness in Rasta-Like Designs: How Far Can We Go?

Application to Pasta

Lorenzo Grassi[1], Fukang Liu[2], Christian Rechberger[3], Fabian Schmid[3], Roman Walch[3,4], and Qingju Wang[5]

[1] Ruhr University Bochum (Germany)
[2] Tokyo Institute of Technology (Japan)
[3] Graz University of Technology (Austria)
[4] TACEO (Austria)
[5] Telecom Paris, Institut Polytechnique de Paris (France)
Lorenzo.Grassi@ruhr-uni-bochum.de, liu.f.ad@m.titech.ac.jp, christian.rechberger@tugraz.at, fabian.schmid@iaik.tugraz.at, walch@taceo.io, qingju.wang@telecom-paris.fr

Abstract. The Rasta design strategy allows building low-round ciphers due to its efficient prevention of statistical attacks and algebraic attacks by randomizing the cipher, which makes it especially suitable for hybrid homomorphic encryption (HHE), also known as transciphering. Such randomization is obtained by pseudorandomly sampling new invertible matrices for each round of each new cipher evaluation. However, naively sampling a random invertible matrix for each round significantly impacts the plain evaluation runtime, though it does not impact the homomorphic evaluation cost. To address this issue, Dasta was proposed at ToSC 2020 to reduce the cost of generating the random matrices.

In this work, we address this problem from a different perspective: How far can the randomness in Rasta-like designs be reduced in order to minimize the plain evaluation runtime without sacrificing the security? To answer this question, we carefully studied the main threats to Rasta-like ciphers and the role of random matrices in ensuring security. We apply our results to the recently proposed cipher Pasta, proposing a modified version called Pasta$_{v2}$ instantiated with one initial random matrix and fixed linear layers – obtained by combining two MDS matrices with the Kronecker product – for the other rounds.

Compared with Pasta, the state-of-the-art cipher for BGV- and BFV-style HHE, our evaluation shows that Pasta$_{v2}$ is up to 100 % faster in plain while having the same homomorphic runtime in the SEAL homomorphic encryption library and up to 30 % faster evaluation time in HElib, respectively.

Keywords: Rasta, Pasta, Pasta$_{v2}$, HHE, Interweaving matrix

1 Introduction

Privacy-preserving cryptographic protocols and primitives, such as homomorphic encryption (HE) and multi-party computation (MPC), have been applied to increasingly more applications in the recent decade. However, applying them to any given use case usually results in a huge performance penalty, both for the runtime of the actual use case and for the communication between the involved parties. Looking at applications involving HE, one can use symmetric ciphers in so-called hybrid homomorphic encryption (HHE) (also called transciphering) [55] to address the large communication overhead between a client encrypting the data and a server performing the homomorphic computations. However, the reduced communication overhead then usually comes at the cost of a larger server runtime overhead, which depends on the symmetric cipher used in HHE. Thus, HHE allows to move the workload from a client to the server and allows embedded devices with less computing power, RAM, and bandwidth to securely outsource computations to a server.

HE-Friendly Schemes. In many HE schemes, such as BFV [10, 28], BGV [11], and CKKS [14], the multiplicative depth of the evaluated circuit still is the main bottleneck due to the absence of an efficient bootstrapping procedure. Consequently, traditional symmetric ciphers, such as AES [19, 21], which were optimized for fast plain performance instead of reducing the multiplicative depth are not well suited for HHE. More recently, the authors in [4] investigate the applicability of the standardized Trivium cipher [12]. While the authors claim an improvement of 2 orders in magnitude, they compare a highly parallelized with a single-threaded implementation, with different levels of maturity in the underlying HE implementation.

Given these problems with traditional encryption schemes, many new symmetric ciphers have been proposed in the literature optimized for HHE minimizing the noise induced by the decryption circuit. These ciphers include LowMC [2], Rasta [24], Dasta [42], Kreyvium [13], FLIP [27], FiLIP [53], Fasta [16], `Elisabeth` [17] (broken in [30]), `HERA` [15] (some versions recently broken in [49]), `Rubato` [40] (version operating over integer rings recently broken in [31]), `Masta` [39], Pasta [25], and more recently YuX [48]. For the HE schemes we target, this means minimizing the multiplicative depth of decryption. Looking at the Benchmarks from [25], ciphers based on the Rasta design strategy are especially well suited for HHE in depth-bounded HE schemes, at the cost of slower plain performance.

Rasta and Rasta-Like Schemes. As the majority of the symmetric schemes in the literature, Rasta and Rasta-like schemes, including Dasta, Fasta, `Masta`, and Pasta, are iterated round function schemes. However, with respect to traditional symmetric cryptographic schemes, they are characterized by the following:

- they are instantiated via new randomly generated affine layers for each new block to be encrypted, ensuring efficient protection against statistical attacks;
- their states have huge sizes for preventing linearization attacks without increasing the number of rounds, and so the depth.

We refer to Section 2 for a recap of the evolution of the Rasta-like designs.

Our Contribution

In this paper, we continue the evolution of the Rasta design strategy and optimize it for better statistical security guarantees and faster plain performance. Especially the latter point is important for the HHE use case since HHE is explicitly designed to remove workload from a client and allow embedded devices to participate in secure outsourcing use cases. Hence, producing a cipher with faster plain encryption runtime further reduces the workload of the beneficiary of the whole HHE pipeline, i.e., the resource-constrained clients.

Minimize the Randomness: From Pasta to Pasta$_{\text{v2}}$. In order to achieve this result, we aim to *minimize the randomness in such designs in a secure way.* We achieve this goal by proposing a new primitive called Pasta$_{\text{v2}}$, which is based on Pasta, but *where only the first affine layer is randomly sampled, and the remaining components are all fixed*, as detailed in Section 3. A detailed security analysis of Pasta$_{\text{v2}}$ is proposed in Section 4. As one may expect, Pasta and Pasta$_{\text{v2}}$ are vulnerable to the same attack vectors, especially the linearization attack. In there, we show that, even with a single random affine layer, Pasta$_{\text{v2}}$ offers the same security as Pasta against such (and other) attack(s).

Moreover, to better understand why this is the optimal strategy for the minimization of randomness, we also analyze the security of variants of such schemes, including the case in which the first affine layer is fixed and any of the remaining components (linear or nonlinear layers) is randomized, as discussed in Section 5.

Remark 1. While there could be some concerns regarding the fact that our new scheme is still secure by only randomizing the first affine layer, we view this as the first step to better understand the Rasta-like design strategy. Specifically, we want to pose the question of whether current Rasta-like ciphers are over-designed with too many random layers. Moreover, studying the security of Pasta$_{\text{v2}}$ can also contribute to a better understanding of the security of Rasta-like ciphers, with particular attention on how much randomness is needed for its security.

Interweaving Matrix: About the Statistical Security of Pasta$_{\text{v2}}$. The security of Pasta$_{\text{v2}}$ is also related to our new generic result proposed in Section 7 regarding its new linear layer. It is defined as a matrix $\mathbb{F}_q^{(n\cdot m)\times(n\cdot m)}$ called *interweaving matrix* obtained by combining two MDS matrices – one over $\mathbb{F}_q^{m\times m}$ and one over $\mathbb{F}_q^{n\times n}$ – via the Kronecker product. There, we prove that the branch number of the obtained matrix is always $n+m$, which refers to the fact that the sum of the numbers of non-zero elements at the input and at the output of an interweaving matrixover $\mathbb{F}_q^{(n\cdot m)\times(n\cdot m)}$ is always at least $n+m$.

This result is also of independent interest due to the large use of the wide-trail design strategy [20], which allows the designers to present a formal argument regarding the security of an SPN construction against linear [52] and differential [7] attacks. As it is well known, the wide-trail strategy aims at designing the round transformation(s) of a symmetric scheme in order to maximize the minimum number of active S-Boxes over multiple rounds. The class of matrices that maximize such parameters is called *Maximum Distance Separable* (MDS). At the current state of the art, a lot of effort has been spent by the community looking for 4×4 (or slightly bigger/smaller) efficient MDS or almost-MDS matrices for designing AES-like schemes. Apart from that, only a few strategies are known for constructing MDS matrices of arbitrary size (as the Cauchy [60] or the Vandermonde [46] matrices).

Our result aims to fill this gap by formally analyzing a way to construct efficient matrices with a reasonable branch number. As a concrete impact, our result could be of broader interest, not only for FHE-friendly designs, but also for MPC- and ZK-friendly symmetric schemes. Since they operate over a huge field (e.g. [1,3,9,26,33–36,44]) and make use of low-degree non-linear functions, these SPN schemes often do not require a linear layer with a maximum branch number in order to achieve security against statistical attacks. The interweaving matrices could be crucial for achieving good performances and security against statistical attacks as well.

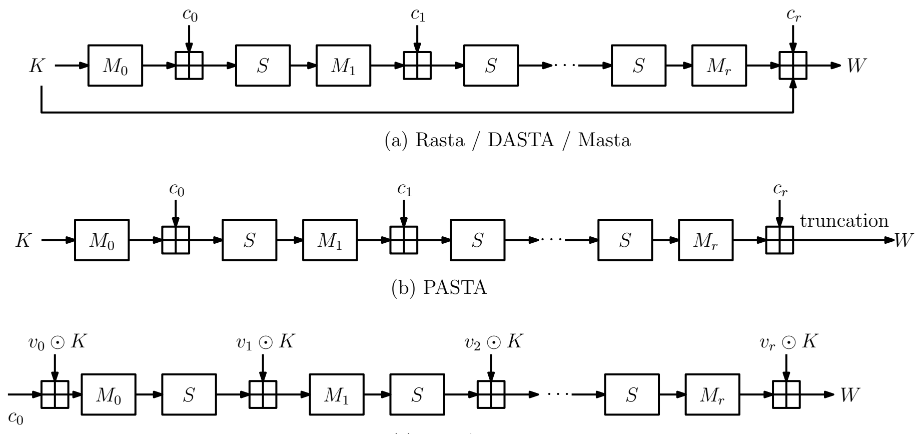
Efficiency of Pasta$_{\mathbf{v2}}$. To show the effectiveness of our proposal, we evaluate its performance in Section 6. While Pasta accounts for special properties of the BGV/BFV HE schemes to be the most efficient symmetric cipher for them to date, its plain performance leaves room for improvements. As shown in [25], encrypting 1.5 MB of data takes 16 s with Pasta compared to only 40 ms with AES.[6] Applying our improvements allows us to reduce the plain encryption time of Pasta by at least 50% depending on the parameters while keeping its advantages for fast HE decryption runtime.

Notation. Let $t \geq 1$. We represent elements of $\mathbb{F}_p^t$ as vectors $x = (x_0, x_1, \ldots, x_{t-1})$. For vectors $x \in \mathbb{F}_p^{2t}$, we denote $x := x_L \| x_R$ where $x_L, x_R \in \mathbb{F}_p^t$ are the left and the right t words, respectively. Further, we write $\texttt{rot}_i(y)$ to indicate a rotation of the vector $y \in \mathbb{F}_p^t$ by i steps to the left. With $y \odot m$, we denote the element-wise product (Hadamard product) between two vectors $y, m \in \mathbb{F}_p^t$. With $\text{diag}(x_1, \ldots, x_t)$ we denote a diagonal matrix of size $t \times t$ whose diagonal is $(x_1, \ldots, x_t)$. Finally, given matrices $M \in \mathbb{F}_p^{m \times m}$ and $N \in \mathbb{F}_p^{n \times n}$, we denote $M \otimes N \in \mathbb{F}_p^{(m \cdot n) \times (m \cdot n)}$ as the Kronecker product of two matrices.

2 Preliminary: Evolution of Rasta-like Primitives

A dedicated symmetric-key primitive for HHE should have a low AND-depth [29, 55]. At CRYPTO 2018, a family of FHE-friendly stream ciphers called Rasta was proposed [24], and it sheds new insight into secure FHE-friendly symmetric-key designs. The main novelty of the Rasta design strategy [24] comes from the realization that a major class of attacks on symmetric ciphers, namely statistical attacks [6–8, 23, 43, 45, 47, 52, 59], depends on a large number of cipher evaluations (i.e., use many plaintext-ciphertext pairs) concerning the same cipher.

[6] While AES is significantly faster in plain, its comparatively huge multiplicative depth would lead to infeasible HHE server runtimes [25].



(a) Rasta / DASTA / Masta

(b) PASTA

(c) HERA

Fig. 1: Constructions of existing Rasta-like primitives.

Consequently, these attacks can be mitigated by randomizing the cipher for each evaluation.

Here, we recall and compare the main existing Rasta-like primitives, which are depicted in Fig. 1. There, c_i, v_i are random vectors, M_i are random matrices, and S denotes the nonlinear layer.

Rasta. In Rasta, both the invertible matrices $M_0, \ldots, M_r$ and the round-constants $c_1, \ldots, c_r$ are randomly generated via an Extendable-Output Function (XOF) [56] seeded with a nonce N and a block counter i. Due to this special structure, algebraic attacks become the main threats, and the most effective one against Rasta is the linearization attack. However, due to the usage of the quadratic χ-transformation [18] for the nonlinear layer of Rasta, faster linearization attack on Rasta can be achieved by exploiting a special property of its inverse [50, 51], though Rasta is still secure against this improved attack.

Dasta. The follow-up work Dasta [42] focuses on the slow instance generation of Rasta. In Rasta, random matrices are sampled until all $(r+1)$ are invertible. To reduce this overhead, Dasta generates the matrices $(M_j)_{0 \le j \le r}$ as follows:

$$\forall j \in [0, r]: \; M_j(x) = M_{f,j} \times \mathcal{P}_j(x),$$

where $M_{f,j}$ is a fixed matrix, while $\mathcal{P}_j$ is a structured bit permutation seeded with the block counter i. In this way, Dasta significantly outperforms Rasta in the plain evaluation by a factor of 200 to 400 [42], while the performance in homomorphic evaluation remains almost the same (The instance generation runtime is negligible compared to the homomorphic key-stream generation). However, as shown in [50], such a way to generate the random matrix impacts the security margin of Dasta, which is much smaller than the one of Rasta against the linearization attack.

Primitives over Prime Fields: Masta, HERA and PASTA. Both Rasta and DASTA are defined over $\mathbb{F}_2$, which makes them less efficient in many FHE applications, though the AND-depths are relatively low. Therefore, new FHE-friendly primitives have been proposed to address this issue. The ciphers Masta, HERA, and PASTA are thus defined over $\mathbb{F}_p$. First, Masta was introduced as a direct application of the Rasta design strategy. The random matrices are now generated from polynomials with coefficients in a prime field rather than $\mathbb{F}_2$. The Masta client side runtime achieved good results, but the scheme is not geared towards HE. Thus, the homomorphic runtime is slow in many settings. For PASTA, the designers proposed a relatively cheap way to generate a random matrix for each round and prove high branch numbers with high probability. Moreover, to prevent efficient attacks on the Rasta-like ciphers as in [50] which exploits the inverse of the nonlinear layer, the designers of PASTA choose to truncate half of the permutation output to get the keystream. The design generally exploits properties of homomorphic computation for fast decompression. However, their matrix sampling method is still slow compared to the other ciphers. In HERA, on the other hand, the matrices of the linear layers are fixed. However, each round, the round keys are randomized by just element-wise multiplying a random vector v_i (sampled from an XOF seeded with a nonce and block counter) to the master key K. For the input to the first nonlinear layer, it is $M_0(v_0 \odot \mathtt{K} + c_0) = M_0(v_0 \odot \mathtt{K}) + M_0(c_0)$ where c_0 is a known constant. Hence, we can interpret it from a different perspective: First, a fixed matrix is multiplied with a randomly generated diagonal matrix. Then, multiply this new matrix with K, and finally, a constant is added. In this sense, its *first* linear layer is also somehow randomized, but in a slightly different way from Masta and PASTA. With a small state size and a limited number of rounds, HERA generates the smallest number of random elements and achieves the best client-side encryption time. However, by exploiting the special feature of the randomized key schedule and the the small state size, an algebraic attack on HERA using multiple collisions in round keys has been proposed in [49], which can successfully peel off the *last* nonlinear layer of HERA and achieve a full-round attack under the same assumption made by the designers. In Section 6, we dive into the specifics of the performance trade-offs of the $\mathbb{F}_p$-HHE ciphers.

3 Description of PASTA$_{\text{v2}}$

In this section, we introduce PASTA$_{\text{v2}}$ as an evolution of the cipher PASTA.

3.1 PASTA

PASTA [25] is a family of stream ciphers proposed at TCHES 2023. Let p be a prime such that $\log_2(p) > 16$ and $\gcd(p-1, 3) = 1$.[7] Given a secret key $\mathtt{K} \in \mathbb{F}_p^{2t}$, a nonce N and a counter i, a PASTA encryption works as follows:

– the message $m \in \mathbb{F}_p^*$ is first parsed as $m = m_0 \| m_1 \| \ldots \| m_n$ with $m_i \in \mathbb{F}_p^t$;

[7] The power map $x \mapsto x^d$ is invertible over $\mathbb{F}_p$ if and only if $\gcd(p-1, d) = 1$.

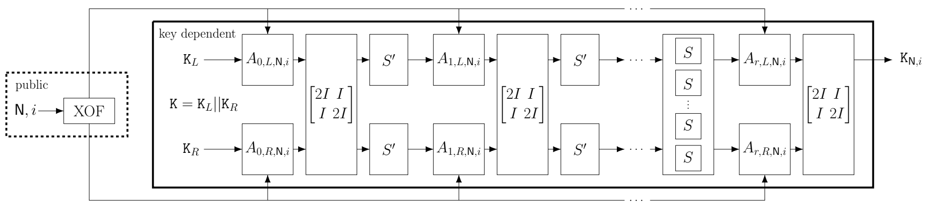


Fig. 2: The truncated r-round Pasta-π permutation to generate the keystream $K_{\mathsf{N},i}$ for block i under nonce N with affine layers $A_{j,k,\mathsf{N},i}$.

- the ciphertext is defined as $c = c_0 \| c_1 \| \cdots \| c_n$ where

$$c_i = m_i + \mathrm{left}_t(\textsc{Pasta}\text{-}\pi(\mathtt{K}, N, i))$$

where Pasta-π is the Pasta permutation described in the following, and where $\mathrm{left}_t(\cdot)$ returns the first t words.

The keystream generation is shown in Fig. 2.

The permutation Pasta-$\pi(x, \mathsf{N}, i)$ on a vector $x \in \mathbb{F}_p^{2t}$ is defined as

$$\begin{aligned}\textsc{Pasta}\text{-}\pi(x, \mathsf{N}, i) = A_{r,\mathsf{N},i} \circ S_{\text{cube}} \circ A_{r-1,\mathsf{N},i} \circ S_{\text{feistel}} \\ \circ A_{r-2,\mathsf{N},i} \circ \ldots \circ A_{1,\mathsf{N},i} \circ S_{\text{feistel}} \circ A_{0,\mathsf{N},i}(x),\end{aligned} \tag{1}$$

where $r \geq 1$ is the number of rounds.

S_{feistel} and S_{Cube} are S-box layers defined as below:

- S_{feistel} is an S-box layer defined as $S_{\text{feistel}}(x) = S'(x_L) \| S'(x_R)$, where S' over $\mathbb{F}_p^t$ is a Feistel structure defined as

$$\forall l \in \{0, 1, \ldots, t-1\}: \qquad (S'(y))_l = \begin{cases} y_l & \text{if } l = 0, \\ y_l + (y_{l-1})^2 & \text{otherwise,} \end{cases} \tag{2}$$

where $y = y_0 \| y_1 \| \cdots \| y_{t-1} \in \mathbb{F}_p^t$, and where $(S'(y))_l$ with the index l refers to the l-th element after applying S'.
- S_{cube} is an S-box defined as $S_{\text{cube}}(x) = x_0^3 \| x_1^3 \| \cdots \| x_{2t-1}^3$.

For each $j \in \{0, \ldots, r\}$, $A_{j,\mathsf{N},i}$ is an affine layer

$$A_{j,\mathsf{N},i}(x) = \begin{bmatrix} 2 \cdot I & I \\ I & 2 \cdot I \end{bmatrix} \begin{bmatrix} M_{j,L,\mathsf{N},i}(x_L) + c_{j,L,\mathsf{N},i} \\ M_{j,R,\mathsf{N},i}(x_R) + c_{j,R,\mathsf{N},i} \end{bmatrix}, \tag{3}$$

where $I \in \mathbb{F}_p^{t \times t}$ is the identity matrix and where $M_{j,L,\mathsf{N},i}, M_{j,R,\mathsf{N},i} \in \mathbb{F}_p^{t \times t}$ and $c_{j,L,\mathsf{N},i}, c_{j,R,\mathsf{N},i} \in \mathbb{F}_p^t$ are generated for each round from an XOF [56] seeded with a nonce N and a counter i.

To efficiently sample each invertible matrix $M_{j,k,\mathsf{N},i} \in \mathbb{F}_p^{t\times t}$, they sample sequential matrices following [37,38]:

$$M_{j,k,\mathsf{N},i} = \left(\tilde{M}_{j,k,\mathsf{N},i}\right)^t, \quad \tilde{M}_{j,k,\mathsf{N},i} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_t \end{bmatrix} \tag{4}$$

for each $k \in \{L, R\}$, where $\alpha_1, \ldots, \alpha_t \in \mathbb{F}_p \setminus \{0\}$. In other words, $M_{j,k,\mathsf{N},i}$ is an invertible matrix which can be built by sampling t random elements and performing $t \cdot (t-1)$ multiplications and $(t-1)\cdot(t-1)$ additions.

3.2 The Birth of Pasta$_{\mathsf{v2}}$

Next, we propose Pasta$_{\mathsf{v2}}$ as a variant of Pasta, in which only the first affine layer is randomized. We keep the main structure the same. Hence, for a prime p such that $\log_2(p) > 16$ and $\gcd(p-1,3) = 1$, the permutation Pasta$_{\mathsf{v2}}$-$\pi(x, \mathsf{N}, i)$ on a vector $x \in \mathbb{F}_p^{2t}$ is defined as

$$\begin{aligned} \text{Pasta}_{\mathsf{v2}}\text{-}\pi(x,\mathsf{N},i) = A_r \circ S_{\text{cube}} \circ A_{r-1} \circ S_{\text{feistel}} \\ \circ A_{r-2} \circ \ldots \circ S_{\text{feistel}} \circ A_1 \circ S_{\text{feistel}} \circ A_{0,\mathsf{N},i}(x), \end{aligned} \tag{5}$$

where $r \geq 1$ is the number of rounds. The differences from Pasta are:

- only the first affine layer $A_{0,\mathsf{N},i}$ is randomized with the seed (N, i);
- the remaining r affine layers $A_1, \ldots, A_r$ are fixed.

Fixed Affine Layers. We use a pre-defined MDS-matrix $M \in \mathbb{F}_p^{t\times t}$ to instantiate the affine layers $A_1, \ldots, A_r$, as shown below:

$$A_j(x) = \begin{bmatrix} 2\cdot I & I \\ I & 2\cdot I \end{bmatrix} \times \begin{bmatrix} M(x_L) + c_{j,L} \\ M(x_R) + c_{j,R} \end{bmatrix},$$

where $I \in \mathbb{F}_p^{t\times t}$ is the identity matrix, and $c_{j,L}, c_{j,R} \in \mathbb{F}_p^t$ are fixed round constants. To sample the MDS matrix $M \in \mathbb{F}_p^{t\times t}$, we instantiate it as a random Cauchy matrix [61]:

$$M_{i,j} = \frac{1}{x_i + y_j},$$

where x_i, y_j are random elements in $\mathbb{F}_p$ such that (i) $x_i \neq x_j$, (ii) $y_i \neq y_j$ and (iii) $x_i + y_j \neq 0$ for each i and j.

As we are going to prove in Section 7, *the branch number of our fixed linear layers A is $t + 2$*. The main differences to the affine layers in Pasta are the following:

Table 1: Two instances of $\textsc{Pasta}_{\mathsf{v2}}$ with 128-bit security (assuming $\log_2(p) > 16$ and $\gcd(p-1,3) = 1$). We emphasize that $\textsc{Pasta}_{\mathsf{v2}}$-3 and $\textsc{Pasta}_{\mathsf{v2}}$-4 have the *same security level*, but a different state-size t implies a different number of rounds (and so depth).

Instance	r	# Key Words $2t$	# Plain/Cipher Words t	XOF
$\textsc{Pasta}_{\mathsf{v2}}$-3	3	256	128	Shake128
$\textsc{Pasta}_{\mathsf{v2}}$-4	4	64	32	Shake128

– the matrix M and the random round constants are chosen during instantiation and *not* sampled from an XOF;
– there is only one matrix M which is the same for both $\textsc{Pasta}$-branches and for each round;
– the matrix M is instantiated to be an MDS matrix.

About the First Affine Layer. The first affine layer $A_{0,\mathsf{N},i}$ is defined in the same way as Eq. (3). In $\textsc{Pasta}_{\mathsf{v2}}$, $c_{0,L,\mathsf{N},i}, c_{0,R,\mathsf{N},i} \in \mathbb{F}_p^t$ are sampled using an XOF seeded with (N, i). The matrices $M_{0,L,\mathsf{N},i}$ and $M_{0,R,\mathsf{N},i}$ are constructed as follows. First of all, we generate two matrices of the form (Eq. (4)) in $\mathbb{F}_p^{t\times t}$ denoted by $M_{f,L}$ and $M_{f,R}$ and fix them, i.e., they will remain the same for different (N, i). Then, using an XOF [56] seeded with (N, i), we sample $2t$ nonzero random elements in $\mathbb{F}_p$, denoted by $(\beta_1, \ldots, \beta_{2t})$. Finally, we define:

$$M_{0,L,\mathsf{N},i} = M_{f,L} \times \mathrm{diag}(\beta_1, \ldots, \beta_t),$$
$$M_{0,R,\mathsf{N},i} = M_{f,R} \times \mathrm{diag}(\beta_{t+1}, \ldots, \beta_{2t}).$$

3.3 Concrete Parameters

Based on our security analysis proposed in the next sections, $\textsc{Pasta}_{\mathsf{v2}}$ requires the same statesize and number of rounds as $\textsc{Pasta}$ for the same security level. Thus, as shown in Table 1, we propose the 3-round and 4-round instances denoted by $\textsc{Pasta}_{\mathsf{v2}}$-3 and $\textsc{Pasta}_{\mathsf{v2}}$-4, respectively. These instances provide at least 128 bits of security (with a security margin of at least 20 % [25]) for the prime fields $\mathbb{F}_p$ with $\log_2(p) > 16$ and $\gcd(p-1,3) = 1$. (We refer to Section 6 for concrete values of p, which depend on the considered application.)

4 Security Analysis of $\textsc{Pasta}_{\mathsf{v2}}$

The security analysis of $\textsc{Pasta}_{\mathsf{v2}}$ is analogous to the one of $\textsc{Pasta}$. Here, we focus on the effect of having only a single random affine layer in the first round. As in $\textsc{Pasta}$, we show that chosen-plaintext attacks do not work. For this reason, we mainly focus on known-ciphertext attacks, i.e., the attacker knows the output of $\textsc{Pasta}_{\mathsf{v2}}$.

About Chosen-Plaintext Attacks. The input of $\textsc{Pasta}_{\mathtt{v2}}$ is composed of a secret key $\mathtt{K}$, a nonce N and a counter i. In particular, (N, i) is set as the input of an XOF, and the output of the XOF is used to construct the first affine layer. In this sense, the attacker cannot control the first affine layer given that the used XOF is secure, even though it can adversarially choose (N, i). On the other hand, $\mathtt{K}$ remains unknown and uncontrolled. Therefore, chosen-plaintext attacks, including differential [7], truncated differential [45], impossible differential [6], and cube [23] attacks cannot work. In more detail, in the case of a differential attack, the difference Δ after the first linear layer is given by

$$\Delta = (M_0 - M_0') \cdot \mathtt{K} + (c_0 - c_0')\,,$$

where M_0, M_0' are random matrices over $\mathbb{F}_p^{2t\times 2t}$ and $c_0, c_0' \in \mathbb{F}_p^{2t}$ are the constant vectors. Since $\mathtt{K}$ is unknown, and since the probability that either (i) $M_0 = M_0'$ or (ii) some rows of M_0 and M_0' are equal is much lower than 2^{-128}, it seems not possible to set up such an attack.

Linear Cryptanalysis. Although the differential attack cannot work, the linear attack [52] can work for Rasta-like ciphers in a slightly different way, as indicated by the designers of $\mathtt{HERA}$ [15], i.e., it can be reduced to solving a Learning with Errors (LWE) [57] like problem. This attack is equivalent to finding a linear approximation of the $\textsc{Pasta}_{\mathtt{v2}}$ permutation holding with a high probability. Based on the fact that the branch number of our fixed linear layers is $t + 2$ (as we are going to prove in Section 7), we have built a simple MILP model as in [54] and found that the minimal number of active nonlinear operations $x \mapsto x^2$ for the first two rounds is 16 and 64 for $\textsc{Pasta}_{\mathtt{v2}}$-4 and $\textsc{Pasta}_{\mathtt{v2}}$-3, respectively. In addition, according to Lemma 1 (Appendix B.5) in [15], the upper bounds of the linear probability for the nonlinear transforms $x \mapsto x^2$ over $\mathbb{F}_p$ is $\frac{1}{p}$. Since $p^{16} > 2^{128}$ for $p > 2^{16}$, we conjecture that $\textsc{Pasta}_{\mathtt{v2}}$ is secure against this attack.

Algebraic Security against Linearization, Gröbner basis and Interpolation Attacks: Relation with $\mathtt{HERA}$. Similar to $\textsc{Pasta}$, the most threatening attack on $\textsc{Pasta}_{\mathtt{v2}}$ is the linearization attack. In such an attack, the attacker can set up many equations in terms of $\mathtt{K}$ according to the outputs of $\textsc{Pasta}_{\mathtt{v2}}$ and solve the equation system with Gaussian elimination by treating each different monomial as an independent variable. The degree denoted by d of these equations is upper bounded by 12 and 24 for $\textsc{Pasta}_{\mathtt{v2}}$-3 and $\textsc{Pasta}_{\mathtt{v2}}$-4, respectively. In this way, the time complexity of the linearization attack is $\binom{2t+d}{d}^{\omega}$ where $2 \le \omega \le 3$. Our choice for the parameters has ensured that this complexity is larger than 2^{128} under $\omega = 2$.

For the above time complexity, we implicitly assume that all polynomial equations in $\mathtt{K}$ describing $\textsc{Pasta}_{\mathtt{v2}}$ are dense, i.e., almost all monomials appear in the final representation. To verify this assumption, we have practically verified the density of the polynomials for $\textsc{Pasta}_{\mathtt{v2}}$. To avoid the effect of cancellations, we used prime numbers larger than 2^{16}. We observe that for the state sizes we tested, the actual number of monomials in the output word with the smallest number of monomials is always very close to the theoretic maximum number of monomials (details in Fig. 5 – App. A), which is also the case for Pasta as shown in [25]. In this sense, $\text{Pasta}_{\texttt{v2}}$ and Pasta provide equivalent security against algebraic attacks, such as linearization, Gröbner basis, and interpolation attacks, whose time complexity is closely related to the degree and density of the polynomial equations.

The crucial point is that the first nonlinear layer can**not** be efficiently peeled off by linearizing it. Specifically, to linearize the first round in this way, it would be necessary to introduce $\binom{2t+2}{2}$ intermediate variables to represent each possible term of degree smaller than 2 formed by the $2t$ key variables, which would not improve the straightforward linearization attack. More details of this type of attack can be referred to Section 5.1. Besides, note that *the particular way in which we construct the first random affine layer of* $\text{Pasta}_{\texttt{v2}}$ *is essentially comparable to the one of* Pasta *or* ***HERA***, where the key is multiplied via a random diagonal affine layer. *Hence, if the first nonlinear layer can be peeled off for* $\text{Pasta}_{\texttt{v2}}$*, we can also peel it off for* Pasta *and* ***HERA****, which will directly lead to a breakthrough in the analysis of these two ciphers. Still, to the best of our knowledge, no attack that peels off the first non-linear layer has been currently proposed against* Pasta *or* ***HERA*** *in the literature, which makes us confident on the security of* $\text{Pasta}_{\texttt{v2}}$ *as well.* (We refer to the next Section 5 for a detailed analysis regarding the impact of the first random affine layer against the linearization attacks.)

Algebraic Attack using Multiple Collisions. Recently, a new algebraic attack on HERA has been proposed [49] by using multiple collisions in the randomized round keys. This attack on some parameters of HERA succeeds because one could efficiently peel off the last nonlinear layer when the cost to find collisions in round keys is low, i.e., when the state size p^t is small. However, the $\text{Pasta}_{\texttt{v2}}$ state size is of about $2t \cdot \log_2 p > 32 \cdot t \geq 1024$ bits, which makes finding collisions in the state too expensive (i.e., much larger than 2^{128}). Therefore, our design is secure against the algebraic attack proposed in [49].

Higher-order Differential Attack. For simplicity, let us denote the fixed permutation after the first affine layer of $\text{Pasta}_{\texttt{v2}}$ by $\mathcal{P}(x) : \mathbb{F}_p^{2t} \mapsto \mathbb{F}_p^t$. In this way, the degree of $\mathcal{P}(x)$ is 12 and 24 for $\text{Pasta}_{\texttt{v2}}$-3 and $\text{Pasta}_{\texttt{v2}}$-4, respectively. Due to such a low degree, one may feel that a higher-order differential attack over the prime fields [5] can be mounted in a different way. However, note that the input x of $\mathcal{P}(x)$ is computed as $x = M_0(\texttt{K}) + c_0$, where both $c_0 \in \mathbb{F}_p^{2t}$ and the matrix M_0 over $\mathbb{F}_p^{2t\times 2t}$ are randomly generated. To mount a higher-order differential attack over the prime field [5], the attacker needs to collect at least 13 inputs $x = (x_1, \ldots, x_{2t})$ such that there exists an index j such that x_j travels in a multiplicative subgroup of $\mathbb{F}_p$, while the remaining $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_{2t})$ takes the same value. Due to the addition of the random vector c_0, the input of $\mathcal{P}(x)$ can be viewed to be random as well. Hence, the probability that there exists a j such that $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_{2t})$ takes the same value for 13 random x is equal to $\frac{t}{(p^{2t-1})^{12}} < 2^{-128}$. Hence, $\text{Pasta}_{\texttt{v2}}$ is secure against this attack.

Meet-in-the-Middle and Guess-and-Determine Attacks. For the fixed permutation $\mathcal{P}(x)$ after the first affine layer, the attacker can make many offline queries to $\mathcal{P}(x)$ and collect many $(x, \mathcal{P}(x))$ tuples. At the online phase, with each random $A_{0,\mathsf{N},i}$, the attacker only knows t words of the output of the fixed permutation. Even if a match is found for these t words of $\mathcal{P}(x')$, the corresponding $2t$ words x' stored in the precomputed table is correct with probability $1/p^t$, which means the time complexity of this attack is lower bounded by p^t. Since $\log_2(p) > 16$ and $t > 32$ in Pasta$_{\mathsf{v2}}$, all the proposed instances have at least 128-bit security against this attack.

Similarly, assume that the attacker guesses the truncated part in order to invert the permutation. Since each Pasta$_{\mathsf{v2}}$ state is composed of $2t$ words and t words are truncated, the time complexity of such a guess-and-determine attack is lower bounded by $\mathcal{O}(p^t)$. Since $\log_2(p) > 16$ and $t > 32$, Pasta$_{\mathsf{v2}}$ is secure against this attack.

5 Design Rationale

In the previous section, we showed that Pasta$_{\mathsf{v2}}$ is secure even with a single randomized layer. Here, we explain the design rationale of this new updated version of Pasta. The design Pasta$_{\mathsf{v2}}$ is the result of our considerations/analysis regarding the following questions:

1. which layer should we randomize to maximize the security?
2. what are the benefits of fixing the other layers from a security point of view?
3. what are the advantages of truncation with respect to feed-forward?

Since truncation is used in Pasta, we refer to Appendix B for a discussion on question (3). here, we focus on questions (1) and (2).

5.1 Choice of the Randomized Layer from a Security Point of View

To minimize the randomness in Rasta-like designs, the optimal choice is obviously only to randomize one layer, i.e., either one linear layer or one nonlinear layer. here, we explain our choice to randomize the first linear layer.

Randomizing Nonlinear Layers. First, can we simply randomize one nonlinear layer? Compared with randomizing the linear layer, using a random nonlinear layer implies the usage of a random S-box. One relatively cheap way to randomize the S-box is to introduce some random constants. For example, for the S-box $S'(y)$ used in Pasta, which is specified in Eq. (2), we can introduce $2(t-1)$ random nonzero elements in $\mathbb{F}_p$ sampled with an XOF seeded with (N, i). Denote these random nonzero elements by $(a_1, \ldots, a_{t-1}, b_1, \ldots, b_{t-1}) \in \mathbb{F}_p^{2t-2}$. Then, this S-box can be randomized as follows:

$$(S'(y))_l = \begin{cases} y_l & \text{if } l = 0, \\ y_l + a_{l-1} \cdot y_{l-1} \cdot (y_{l-1} + b_{l-1}) & \text{otherwise.} \end{cases}$$

Similarly, given $4t$ non-zero random elements $(a'_0, \ldots, a'_{2t-1}, b'_0, \ldots, b'_{2t-1})$ in $\mathbb{F}_p$, S_{cube} an be randomized as follows:

$$S_{\text{cube}}(x) = a'_0 \cdot (x_0 + b'_0)^3 \| \cdots \| a'_{2t-1} \cdot (x_{2t-1} + b'_{2t-1})^3 ,$$

However, even if all nonlinear layers in Pasta are randomized in this way, this choice impacts the security level of the scheme, which is reduced due to the linearization attack. Specifically, since the first affine layer remains the same for different (N, i), let us introduce the equivalent key $\mathtt{K}' = (\mathtt{K}'_0, \ldots, \mathtt{K}'_{2t-1})$ satisfying

$$\mathtt{K}' = A_0(\mathtt{K}),$$

where A_0 is the first affine layer. Hence, the attacker can skip the first fixed linear layer and look for $\mathtt{K}'$ directly. Next, for the above way to randomize S_{feistel}, let us further introduce $2(t-1)$ variables $(\mathtt{K}''_0, \ldots, \mathtt{K}'_{2t-3})$ defined as

$$\begin{aligned} \forall i \in [0, t-2] : \qquad & \mathtt{K}''_i = (\mathtt{K}'_{i+1})^2 , \\ \forall i \in [t-1, 2t-3] : \qquad & \mathtt{K}''_i = (\mathtt{K}'_{i+2})^2 . \end{aligned}$$

No matter what (N, i) are, the state after the first round will always be linear in the $2t + 2t - 2 = 4t - 2$ variables $(\mathtt{K}'_0, \ldots, \mathtt{K}'_{2t-1})$ and $(\mathtt{K}''_0, \ldots, \mathtt{K}''_{2t-3})$.

As a result, for 3-round Pasta and for each different (N, i), the attacker can set up t equations in $4t-2$ variables whose degree is only $2 \times 3 = 6$. This means that the attacker only needs to collect $\sum_{i=0}^{6} \binom{4t-2+i-1}{i} = \binom{4t-2+6}{6}$ such equations with about $\binom{4t-2+6}{6}/t$ different (N, i) to solve these variables with the linearization technique. The time complexity is upper bounded by $\binom{4t-2+6}{6}^{\omega}$, where $2 < \omega \leq 3$. Usually, from the perspective of designers, $\omega = 2$ is chosen. Similarly, for this new version of 4-round Pasta, the time complexity to break it is upper bounded by $\binom{4t-2+12}{12}^{\omega}$.

For comparison, the time complexity to break 3 and 4 rounds of the original Pasta is upper bounded by $\binom{2t+12}{12}^{\omega}$ and $\binom{2t+24}{24}^{\omega}$, respectively, because the attackers cannot efficiently peel off the first nonlinear layer with this method due to the first random affine layer. Such a property also holds for $\text{Pasta}_{\mathtt{v2}}$. More specifically, although they can also introduce $2t-2$ intermediate variables to linearize the first nonlinear layer for each different (N, i), these intermediate variables are different for different (N, i) since the first affine layer varies for different (N, i), that is, they will have different relations with the secret key $\mathtt{K}$ for different (N, i). This is equivalent to the fact that the first round cannot be efficiently peeled off, or $\binom{2t+2}{2}$ intermediate variables should be introduced to linearize the first round.

Randomizing an Affine Layer in the Middle Rounds. Let's consider r rounds of Pasta. If the affine layer A_i (for $i > 0$) is the only one to be randomized, the attacker only needs to recover the input state of A_i, since such state remains the same for different (N, i). Once it is found, it is possible to compute backward to recover the secret key. In this way, attacking r-round Pasta is reduced to attacking $r - i$ rounds of Pasta, which significantly reduces the security of Pasta.

5.2 Benefits of Having Fixed Layers from a Security Point of View

As stated Section 4, the input of the fixed permutation $P(x)$ after the first affine layer is random. If $P(x)$ is malicious, e.g., instantiated with weak affine layers, it may still weaken the security of $\text{PASTA}_{\texttt{v2}}$, allowing for concrete attacks that aim to recover the input of $P(x)$. By choosing secure affine layers in the fixed permutation, e.g., using MDS matrices, it is possible to avoid this scenario and have a better security argument.

Specifically, the fixed affine layers in $\text{PASTA}_{\texttt{v2}}$ provide full diffusion after only one round and have a branch number of $t + 2$, as we are going to prove in Section 7. Comparing this to PASTA, its branch number is only shown to be larger than $t/2$ with a high probability. In summary, our changes allow us to keep all the advantages of randomizing the cipher while fully eliminating the possibility of weak matrices, which might compromise security in some instances. For example, there are many instances of weak matrices making the MPC-friendly cipher LowMC vulnerable to the interpolation attack [22, Sect. 1]: "*[...] the designers of LowMC allow to instantiate it using a pseudo-random source that is not cryptographically secure. Our result shows that this is risky, as using an over-simplified source for pseudo-randomness [...] allow finding weak instances [...]*".

6 Benchmarks

In this section, we report on the practical performance of $\text{PASTA}_{\texttt{v2}}$ and the consequences of only randomizing the first affine layer. We compare our implementation with the $\mathbb{F}_p$ ciphers outlined in Section 2 based on a benchmarking framework[8] provided by [25]. We separate our performance evaluation into plain encryption and homomorphic evaluation times and evaluate our benchmarks with 33-bit plaintext primes. The plaintext prime p defines the plaintext domain $\mathbb{F}_p$. Increasing p has a negative impact on HE performance. Next to an overview of the HE parameters, we present further tests, including different primes, in Appendix D. First, we reiterate the findings of previous performance comparisons. In [25], the authors show that their HHE *cipher* PASTA *is the fastest in the* HE *domain with the 4 round version being the best for few low-precision numbers and 3 round version being the best when applied to bigger use cases.* The advantages are two-fold:

- a relatively fast homomorphic decompression, and
- a small number of rounds.

Having fewer rounds reduces the noise impact in HE computation and allows for smaller, more efficient parameter settings. Having more efficient parameters positively impacts the performance of subsequent use case evaluations. On the other side, for performance on the client side, 3-round PASTA is outperformed by all competitors, while the 4-round version is better but still far off compared to HERA.

[8] https://github.com/IAIK/hybrid-HE-framework/

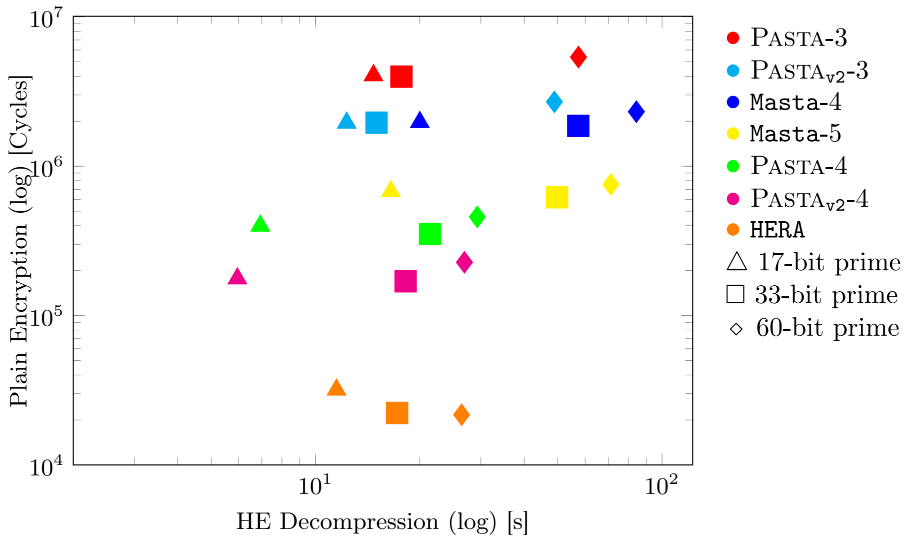


Fig. 3: Comparison of the different $\mathbb{F}_p$ ciphers. The two metrics are the runtime in the homomorphic domain versus the CPU cycles for client-side encryption. All ciphers are indicated with colors and tested with different plaintext prime sizes indicated with shapes.

Plain Benchmarks. In Table 2, we report the performance numbers for the most prominent $\mathbb{F}_p$ HHE ciphers. This table depicts the runtime of the encryption implementation with a 33-bit prime in CPU cycles averaged over 1000 times. This metric allows us to show more minor differences in runtime and be independent of the hardware used. We will discuss the HE benchmarks on the right side later. With different plaintext primes, the numbers change, but the overall relations between the ciphers remain the same. The plain performance section shows the total cycles needed for the entire encryption process, then separated into nonce-dependent instance generation and the computation of the key stream.

HERA is still the fastest cipher when evaluated in plain. On the one hand, Pasta$_{v2}$ roughly halves the gap towards it. On the other hand, it retains the advantages in the HE domain with similar speed for single blocks and lower noise costs. In HE computation, higher noise costs are compensated with less efficient parameter settings. We present practical consequences in Appendix D.

The other ciphers show a clear trade-off between round number and state size performance. Smaller state sizes reduce the number of random elements, reducing the high cost for the client side. Additional rounds, however, hurt HE evaluation, sometimes enforcing less efficient parameter settings. Within this trade-off, we compare the lower round instances of Pasta, Pasta$_{v2}$, and Masta and their higher round instances separately. For the higher round versions, Pasta outperformed Masta in both domains and Pasta$_{v2}$ further increases the gap by doubling the plain encryption speed. Given the low-round versions, Masta was

Table 2: Detailed performance metrics. Plain encryption time is split into nonce-dependent instance generation and generation of the key stream. HE performance shows runtime for encrypting the secret key and decompressing ciphertexts.

	Plain Performance [Cycles]			HE Performance [s]	
Cipher	Total	Instance Generation	Encrypting	Enc. key	Decomp.
Masta-4	1 862 325	712 804	1 149 521	0.095	57.3
Masta-5	619 314	262 892	356 422	0.096	49.8
HERA	**22 294**	**13 607**	**8 687**	0.108	17.2
Pasta-3	3 978 645	2 312 924	1 665 721	0.037	17.7
Pasta-4	351 994	250 693	101 301	0.099	21.4
Pasta$_{v2}$-3	1 956 656	49 645	1 907 011	**0.033**	**15.0**
Pasta$_{v2}$-4	169 632	14 353	155 279	0.094	18.2

two to three times faster than Pasta on the client side Pasta$_{v2}$ closes this gap significantly down to 0-20% slower depending on the plaintext prime.

Having roughly twice the speed of Pasta follows from the new design. First, we note that since Pasta$_{v2}$ has the same number of rounds and the same S-boxes as Pasta, their multiplicative depth remains the same. To successfully encrypt one block with statesize t, Pasta requires to sample $4 \cdot t$ random elements per affine layer ($2t$ for the round constants and t for each of the two submatrices), hence, a total of $4 \cdot t \cdot (r+1)$ random elements. Pasta$_{v2}$, on the other hand, only has one random affine layer. Hence, it is enough to sample $4 \cdot t$ elements, which is significantly smaller than required for Pasta. Further, in Pasta, these random words must be transformed into the $t \times t$ matrices, which comprise each affine layer branch. Each matrix generation requires $t \cdot (t-1)$ field multiplications and $(t-1) \cdot (t-1)$ field additions. On the contrary, in Pasta$_{v2}$, the random words are only applied to the matrix's main diagonal in the first round, resulting in t field multiplications.

After the nonce-dependent instance is created, encryption timings are very similar for both Pasta and Pasta$_{v2}$. Given the slightly different implementations for measuring the instance generation, the encryption time of Pasta is slightly faster. Overall, the speedup of instance generation far outweighs the minor drawbacks in encryption time.

Homomorphic Benchmarks. To show the effect of our changes on the runtime of HHE use cases, we redo the benchmarks from [25] using their open-source benchmarking framework.[9] Hence, we benchmark the runtime of homomorphically evaluating the decryption circuit for one block of data (i.e., homomorphic decompression).

We give benchmarks for two state-of-the-art HE libraries, namely SEAL [58] and HElib [41], which implement the BFV [10, 28] and BGV [11] HE schemes,

[9] We run all benchmarks on a Linux server with an AMD Ryzen 9 7900X CPU (4.7 GHz). Each benchmark only has access to one thread. The source code is available at `https://github.com/IAIK/hybrid-HE-framework`

respectively. The homomorphic decompression implementations of Pasta and $\text{Pasta}_{\texttt{v2}}$ have the following difference: The matrix in the affine layers is the same for each round (after the first one) in $\text{Pasta}_{\texttt{v2}}$ but different in Pasta, one does not have to encode this matrix every round into HE plaintext polynomials when using $\text{Pasta}_{\texttt{v2}}$. Similar or slightly cheaper results are expected for $\text{Pasta}_{\texttt{v2}}$, depending on the HE libraries implementation of encoding.

SEAL Benchmarks. It is found that our changes for Pasta barely affect the benchmarks in the SEAL library. Consequently, less matrix encodings have no real effect in SEAL and lead to practically equivalent benchmarks when using Pasta and $\text{Pasta}_{\texttt{v2}}$. We provide detailed benchmarks for the SEAL library in Appendix D.2.

HElib Benchmarks. In Table 2, we evaluate the HE decompression and symmetric key encryption with different instances of the $\mathbb{F}_p$ HHE ciphers in a 33-bit prime field $\mathbb{F}_p$. We refer to Appendix D.3 for benchmarks with two other prime fields and a more extensive discussion of HE parameters. Most importantly, for the benchmarks discussed, we selected parameters that provide the necessary noise budget and a security parameter λ' such that $\lambda' \geq 128$ bit. Contrary to the benchmarks in the SEAL library, our changes significantly impact the homomorphic decompression runtime in the HElib library. This speedup stems from a more expensive matrix encoding in HElib. Fewer matrices have to be encoded in $\text{Pasta}_{\texttt{v2}}$ since each affine layer (except the first one) uses the same matrix. Depending on the parameters, these changes lead to a runtime advantage in the range of 10 % to 30 %. Regarding noise budget, fluctuations in the range of 1 bit can be observed, which are most likely caused by random Gaussian noise samples.

Discussion. In general, Pasta can be seen as current state-of-the-art in HHE for the BFV and BGV HE schemes. Pasta outperforms `Masta` in the HE domain. Compared to `HERA`, noise consumption and multiplicative complexity ultimately make it the better choice in many applications. We further display these scenarios in Appendix D. Given these preconditions, $\text{Pasta}_{\texttt{v2}}$ is a straight improvement over Pasta. Firstly, $\text{Pasta}_{\texttt{v2}}$ requires fewer random words, significantly improving the client-side encryption. Given the BGV scheme and expensive matrix encoding, $\text{Pasta}_{\texttt{v2}}$ outperforms Pasta in HElib. Finally, the multiplicative complexity is the same s.t. all parameter settings in the HE domain also apply. In Fig. 3, we see clearly that $\text{Pasta}_{\texttt{v2}}$ outperforms Pasta in the plain and HE domains. Further, the concrete choice of p is determined by the concrete HE use case. It should be set as the lowest possible value that does not lead to unwanted overflows in the arithmetic computations. Finally, in the case of a heavily constrained client `HERA` might be the optimal choice depending on the concrete capabilities. However, as mentioned before, higher multiplicative depth diminishes the advantages seen in Fig. 3, the results presented for the 60-bit prime were computed with an HE security parameter of $\lambda' = 89$-bit as increasing security would increase runtime significantly.

7 Branch Number of an Interweaving Matrix

As a final result, we analyze and generalize the design strategy used to set up the internal matrices of PASTA and PASTA$_{\mathtt{v2}}$. The idea is to construct $(n\cdot m)\times(n\cdot m)$ invertible matrices whose branch number is equal to $n+m$ by combining two MDS matrices of dimensions $n\times n$ and $m\times m$. We call the result an "*interweaving matrix*". This strategy is not new in the literature since it is already used in GRIFFIN, `HERA`, and `Rubato`, as discussed later. Still, a formal analysis is missing. here, we aim to fill this gap.

Definition of an Interweaving Matrix. We start by defining an interweaving matrix and continue with proving its branch number.

Definition 1. *Given an $m \times m$ MDS matrix M and an $n\times n$ MDS matrix N, we define the interweaving matrix $Z \in \mathbb{F}_q^{(m\cdot n)\times(m\cdot n)}$ as the Kronecker product of the matrices M and N, given by*

$$Z := M \otimes N = \begin{bmatrix} M_{0,0}\cdot N & M_{0,1}\cdot N & \dots & M_{0,m-1}\cdot N \\ M_{1,0}\cdot N & M_{1,1}\cdot N & \dots & M_{1,m-1}\cdot N \\ \vdots & & \ddots & \vdots \\ M_{m-1,0}\cdot N & M_{m-1,1}\cdot N & \dots & M_{m-1,m-1}\cdot N \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} M_{0,0}\cdot I & M_{0,1}\cdot I & \dots & M_{0,m-1}\cdot I \\ M_{1,0}\cdot I & M_{1,1}\cdot I & \dots & M_{1,m-1}\cdot I \\ \vdots & & \ddots & \vdots \\ M_{m-1,0}\cdot I & M_{m-1,1}\cdot I & \dots & M_{m-1,m-1}\cdot I \end{bmatrix} \times \begin{bmatrix} N & 0 & \dots & 0 \\ 0 & N & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & N \end{bmatrix}, \tag{7}$$

where $I \in \mathbb{F}_q^{n\times n}$ is the identity matrix.

Invertibility of an Interweaving Matrix. We first recall that an arbitrary matrix $M \in \mathbb{F}_q^{m\times m}$ is *invertible* if and only if its determinant $\det(M)$ is co-prime with q, i.e., $\gcd(\det(M), q) = 1$. The determinant of a Kronecker product is given by $\det(M\otimes N) = \det(M)^n \cdot \det(N)^m$. Given that both N and M are MDS matrices (as by Definition 1) and hence invertible, it follows that $\gcd(\det(M \otimes N), q) = 1$. Thus, an interweaving matrix is always invertible.

An Alternative Representation of an Interweaving Matrix. Roughly speaking, multiplying a vector $\boldsymbol{x} \in \mathbb{F}_q^{n\cdot m}$ by an interweaving matrix Z corresponds to do the following:

1. re-arrange the vector $\boldsymbol{x} \in \mathbb{F}_q^{n\cdot m}$ into a 2-dimensional vector (or matrix) in $\mathbb{F}_q^{m\times n}$ of n rows and m columns;
2. multiply each column of this 2-dimensional vector with the $n \times n$ matrix N;
3. multiply each row of the obtained 2-dimensional vector with the $m \times m$ matrix M.

In short: $\boldsymbol{x} \xrightarrow{\text{MixColumns}} N \times \boldsymbol{x} \xrightarrow{\text{MixRows}} (N \times \boldsymbol{x}) \times M^T$, where $\boldsymbol{x}$ is the 2-dimensional vector. More formally, denoting that the diagonal matrix $\text{diag}(A)_x :=$

$\text{diag}(A,\ldots,A)_x$ indicates x matrices A in the diagonal, we can further define a shuffle $\Sigma^{(x)} \in \mathbb{F}_q^{(m\cdot n)\times(m\cdot n)}$ as

$$\Sigma^{(x)} := (\text{diag}(\boldsymbol{e}_0^{(x)})_y, \text{diag}(\boldsymbol{e}_1^{(x)})_y, \ldots, \text{diag}(\boldsymbol{e}_{x-1}^{(x)})_y)^{\mathbf{T}},$$

where $(x,y) = (m,n)$ or (n,m), $\mathbf{0}^{(x)} = (0,0,\ldots,0) \in \mathbb{F}_q^x$ and $\boldsymbol{e}_i^{(x)} \in \mathbb{F}_q^x$ is the vector which contains zero apart from the i-th element which is equal to 1.

Lemma 1. *Let $Z \in \mathbb{F}_q^{(m\cdot n)\times(m\cdot n)}$ be defined as in Definition 1. It can be re-written as*

$$Z = \Sigma^{(m)} \times \mathit{diag}(M)_n \times \Sigma^{(n)} \times \mathit{diag}(N)_m. \tag{8}$$

The proof is given in App. C.1.

Branch Number of an Interweaving Matrix. First, we recall that the *branch number* of a matrix M is defined as

$$\mathcal{B}(M) := \min_{\boldsymbol{a}\in\mathbb{F}_q^{m\times m}\setminus\{\mathbf{0}\}} \{\text{hw}(\boldsymbol{a}) + \text{hw}(M\times\boldsymbol{a})\},$$

where $\text{hw}(\cdot)$ is the Hamming weight of vector $\boldsymbol{a}$ and is defined as the number of nonzero elements. The branch number of an MDS matrix is $m+1$. Next, we compute the branch number of an interweaving matrix.

Theorem 1. *Let $Z \in \mathbb{F}_q^{(m\cdot n)\times(m\cdot n)}$ be as in Eq. (6). If M and N are both MDS matrices (as required in Definition 1), then its branch number is $n+m$.*

Proof. Given an input $\boldsymbol{x} = (\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{m-1}) \in \mathbb{F}_q^{n\cdot m}$ with α non-zero $\mathbb{F}_q$-words, we compute the minimum number of non-zero $\mathbb{F}_q$-words of the output $Z\times\boldsymbol{x} = \boldsymbol{y} = (\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{n-1}) \in \mathbb{F}_q^{n\cdot m}$ using the representation given in Eq. (6). For this goal, we first remove the final shuffle $\Sigma^{(m)}$, since it does not change the number of active words. We further define intermediate variables $\boldsymbol{u} = (\boldsymbol{u}_0,\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{m-1}), \tilde{\boldsymbol{u}} = (\tilde{\boldsymbol{u}}_0,\tilde{\boldsymbol{u}}_1,\ldots,\tilde{\boldsymbol{u}}_{n-1}) \in \mathbb{F}_q^{m\cdot n}$ as the input and output of the shuffle $\Sigma^{(n)}$.

One Active $\mathbb{F}_q^n$-Word Case. Let's start by considering the case in which the input $\boldsymbol{x}$ contains at most α non-zero $\mathbb{F}_q$-words with the indices $i_0, i_1, \ldots, i_{\alpha-1}$ of the active (i.e., non-zero) $\mathbb{F}_q$-words in a set $S_j^{(n)} = \{j\cdot n, j\cdot n+1, \ldots, j\cdot n+n-1\}$ for a certain $j \in \{0,1,\ldots,m-1\}$. Without loss of generality, we assume $i_0 < i_1 < \cdots < i_\alpha$ (see Fig. 4).

We follow the notations in Lemma 1, i.e., $\text{diag}(N)_m = \text{diag}(N,N,\ldots,N)_m$ and $\text{diag}(M)_n = \text{diag}(M,M,\ldots,M)_n$. Based on the representation of Z given in Eq. (8), after the application of $\text{diag}(N)_m$, the number of active $\mathbb{F}_q$-words β is

$$1 \le n+1-\alpha \le \beta \le n\,.$$

Indeed, $n+1-\alpha \le \beta$ since the matrix N is MDS, and $\beta \le n$ follows from the fact that only one matrix N is active. In particular, the indices $k_0 < k_1 < \ldots < k_{\beta-1}$ of the active (i.e., non-zero) $\mathbb{F}_q$-words after the application of N are still in the same set $S_j^{(n)}$ for the same index $j \in \{0,1,\ldots,m-1\}$ as before.

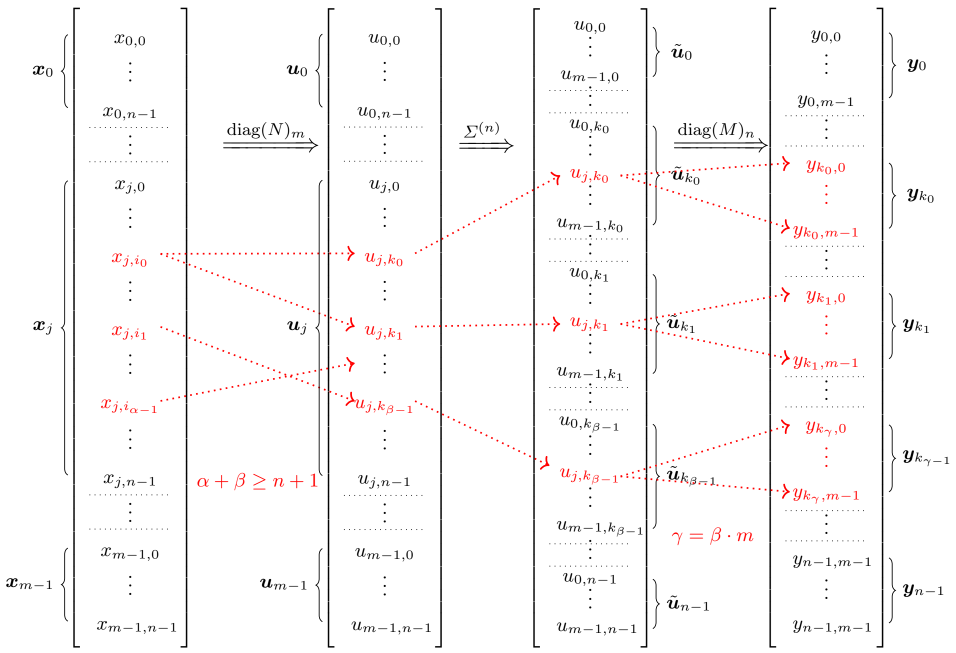


Fig. 4: Branch number for one active $\mathbb{F}_q^n$-word case.

Next, the matrix $\Sigma^{(n)}$ is applied. The number of active $\mathbb{F}_q$-words does not change. However, we observe the following. For each $l \in \{0, 1, \ldots, \beta-1\}$, there exists a $j \in \{0, 1, \ldots, n-1\}$ such that $i_l \in S_j^{(m)} = \{j \cdot m, j \cdot m+1, \ldots, j \cdot m+m-1\}$. Then:

$$\forall h \in \{0, 1, \ldots, \beta-1\} \setminus \{l\} : \qquad i_h \notin S_j^{(m)} \, ,$$

that is, after shuffle $\Sigma^{(n)}$, elements in any $\mathbb{F}_q^n$-word of $\boldsymbol{u}$ cannot appear in the same $\mathbb{F}_q^m$-word of $\tilde{\boldsymbol{u}}$.

In such a case, after the application of $\mathrm{diag}(M)_n$, the number γ of active $\mathbb{F}_q$-words are

$$\gamma = \beta \cdot m \geq (n+1-\alpha) \cdot m \, .$$

Indeed, $\beta \cdot m \leq \gamma$ is due to the facts that (i) β $\mathbb{F}_q^m$-words are active, (ii) each of such words contains only one active $\mathbb{F}_q$-word, and (iii) M is MDS (hence, its branch number is $m+1$). It follows that if one $\mathbb{F}_q$-word is active in an $\mathbb{F}_q^m$-word, all m $\mathbb{F}_q$-words are active after the application of the matrix M (see the last step in Fig. 4).

It follows that the number of active words in inputs and in outputs is at least

$$\alpha + \gamma \geq (n+1-\alpha) \cdot m + \alpha \, .$$

This number is minimized by choosing $\alpha = n$, which implies that the minimum number of active $\mathbb{F}_q$-words is $n + m$, as claimed.

Other Cases. In order to finish the proof, we have to consider the cases in which we have more than a single active $\mathbb{F}_q^n$-word in the input. Since the strategy to analyze this case is equivalent to the one just proposed, and due to the page limit, we present the details in App. C.2. In there, we show that also for this case, the minimum number of active words in inputs and outputs is $n + m$.

$\square$

Interweaving Matrix in Griffin, HERA, and Rubato. Finally, we point out that interweaving matrices are already used in the literature. For example, the matrix used in the ZK-friendly scheme Griffin [32] can also be seen as an interweaving matrix for some parameters, and the branch number following our proof matches the branch number given by the designers of Griffin. The matrix in the linear layers of Griffin [32] for statesizes $t = 4 \cdot t' \geq 8$ is defined as $\text{circ}(2I, I, \ldots, I) \times \text{diag}(N, N, \ldots, N)_{t'}$, where $I \in \mathbb{F}_q^{4\times 4}$ is the identity matrix, and $N = \text{circ}(3, 2, 1, 1)$ is an 4×4 MDS matrix. Since both $\text{circ}(2, 1)$ and $\text{circ}(2, 1, 1)$ are MDS matrices, the final Griffin matrix can be seen as an interweaving matrix when $t' = 2$ or $t' = 3$. Thus, following our proof, the matrices have a branch number of $t' + 4$, which is 6 and 7, respectively. This matches the proof given in [32].

In a similar way, the linear layer used in the HE-friendly schemes HERA [15] and Rubato [40] corresponds to an interweaving matrix. Indeed, each row of the $\mathbb{F}_q^{v^2}$ state of HERA and Rubato is first multiplied by a $v \times v$ MDS matrix, and then each column of the obtained state is multiplied by another $v \times v$ MDS matrix. Based on the result just given, we can easily deduce that the branch number of the $(v \cdot v) \times (v \cdot v)$ interweaving matrix of HERA and Rubato is $2 \cdot v$ which also matches the proofs by the designers.

Acknowledgments. We extend our gratitude to the reviewers of the SAC workshop for their invaluable feedback and insightful comments, which greatly enhanced the quality of this paper. Lorenzo Grassi is supported by the German Research Foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa – 39078197. Fukang Liu is supported by JSPS KAKENHI Grant Numbers JP22K21282, JP24K20733. He is also funded by the commissioned research (No. JPJ012368C05801) by National Institute of Information and Communications Technology (NICT). Qingju Wang is supported by the ANR through project ANR-23-CE39-0009 TRUST. Finally, this work is partly supported by the European Union under the project Confidential6G with Grant agreement ID: 101096435.

References

1. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: ASIACRYPT. LNCS, vol. 10031, pp. 191–219 (2016)

2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: EUROCRYPT. LNCS, vol. 9056, pp. 430–454 (2015)
3. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. IACR Trans. Symmetric Cryptol. **2020**(3), 1–45 (2020)
4. Balenbois, T., Orfila, J., Smart, N.P.: Trivial transciphering with trivium and TFHE. In: Brenner, M., Costache, A., Rohloff, K. (eds.) WAHC. pp. 69–78. ACM (2023)
5. Beyne, T., Canteaut, A., Dinur, I., Eichlseder, M., Leander, G., Leurent, G., Naya-Plasencia, M., Perrin, L., Sasaki, Y., Todo, Y., Wiemer, F.: Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In: CRYPTO. LNCS, vol. 12172, pp. 299–328 (2020)
6. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: EUROCRYPT. LNCS, vol. 1592, pp. 12–23 (1999)
7. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: CRYPTO. LNCE, vol. 537, pp. 2–21 (1990)
8. Bogdanov, A., Wang, M.: Zero Correlation Linear Cryptanalysis with Reduced Data Complexity. In: FSE. LNCS, vol. 7549, pp. 29–48 (2012)
9. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Salen, R., Velichkov, V., Willems, D.: New design techniques for efficient arithmetization-oriented hash functions: ttanemoi permutations and ttjive compression mode. In: CRYPTO. LNCS, vol. 14083, pp. 507–539. Springer (2023)
10. Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In: CRYPTO. LNCS, vol. 7417, pp. 868–886 (2012)
11. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS. pp. 309–325. ACM (2012)
12. Cannière, C.D.: Trivium: A stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC. LNCS, vol. 4176, pp. 171–186. Springer (2006)
13. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. In: FSE. LNCS, vol. 9783, pp. 313–333 (2016)
14. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic Encryption for Arithmetic of Approximate Numbers. In: ASIACRYPT. LNCS, vol. 10624, pp. 409–437 (2017)
15. Cho, J., Ha, J., Kim, S., Lee, B., Lee, J., Lee, J., Moon, D., Yoon, H.: Transciphering Framework for Approximate Homomorphic Encryption. In: ASIACRYPT. LNCS, vol. 13092, pp. 640–669 (2021)
16. Cid, C., Indrøy, J.P., Raddum, H.: FASTA - A Stream Cipher for Fast FHE Evaluation. In: CT-RSA. LNCS, vol. 13161, pp. 451–483 (2022)
17. Cosseron, O., Hoffmann, C., Méaux, P., Standaert, F.: Towards Globally Optimized Hybrid Homomorphic Encryption - Featuring the Elisabeth Stream Cipher. IACR Cryptol. ePrint Arch. p. 180 (2022)
18. Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995), http://jda.noekeon.org/
19. Daemen, J., Rijmen, V.: Rijndael for AES. In: AES Candidate Conference. pp. 343–348. National Institute of Standards and Technology, (2000)
20. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: IMACC. LNCS, vol. 2260, pp. 222–238. Springer (2001)

21. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
22. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized Interpolation Attacks on LowMC. In: ASIACRYPT. LNCS, vol. 9453, pp. 535–560 (2015)
23. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: EUROCRYPT. LNCS, vol. 5479, pp. 278–299 (2009)
24. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In: CRYPTO. LNCS, vol. 10991, pp. 662–692 (2018)
25. Dobraunig, C., Grassi, L., Helminger, L., Rechberger, C., Schofnegger, M., Walch, R.: Pasta: A Case for Hybrid Homomorphic Encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(3), 30–73 (2023)
26. Dobraunig, C., Kales, D., Rechberger, C., Schofnegger, M., Zaverucha, G.: Shorter signatures based on tailor-made minimalist symmetric-key crypto. In: CCS. pp. 843–857. ACM (2022)
27. Duval, S., Lallemand, V., Rotella, Y.: Cryptanalysis of the FLIP Family of Stream Ciphers. In: CRYPTO (1). LNCS, vol. 9814, pp. 457–475 (2016)
28. Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption. IACR Cryptol. ePrint Arch. **2012**, 144 (2012)
29. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic Evaluation of the AES Circuit. In: CRYPTO. LNCS, vol. 7417, pp. 850–867. Springer (2012)
30. Gilbert, H., Boissier, R.H., Jean, J., Reinhard, J.: Cryptanalysis of Elisabeth-4. In: ASIACRYPT (3). LNCS, vol. 14440, pp. 256–284 (2023)
31. Grassi, L., Ayala, I.M., Hovd, M.N., Øygarden, M., Raddum, H., Wang, Q.: Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on Rubato. In: CRYPTO (3). LNCS, vol. 14083, pp. 305–339 (2023)
32. Grassi, L., Hao, Y., Rechberger, C., Schofnegger, M., Walch, R., Wang, Q.: Horst Meets Fluid-SPN: Griffin for Zero-Knowledge Applications. In: CRYPTO. LNCS, vol. 14083, pp. 573–606. Springer (2023)
33. Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M., Walch, R.: Reinforced concrete: A fast hash function for verifiable computation. In: SIGSAC. pp. 1323–1335. ACM (2022)
34. Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M., Walch, R.: Monolith: Circuit-friendly hash functions with new nonlinear layers for fast and constant-time implementations. IACR Cryptology ePrint Archive **2023**, 1025 (2023)
35. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. In: USENIX. pp. 519–535. USENIX Association (2021)
36. Grassi, L., Lüftenegger, R., Rechberger, C., Rotaru, D., Schofnegger, M.: On a generalization of substitution-permutation networks: The HADES design strategy. In: EUROCRYPT. LNCS, vol. 12106, pp. 674–704. Springer (2020)
37. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: CRYPTO. LNCS, vol. 6841, pp. 222–239 (2011)
38. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: CHES. LNCS, vol. 6917, pp. 326–341 (2011)
39. Ha, J., Kim, S., Choi, W., Lee, J., Moon, D., Yoon, H., Cho, J.: Masta: An HE-Friendly Cipher Using Modular Arithmetic. IEEE Access **8**, 194741–194751 (2020)
40. Ha, J., Kim, S., Lee, B., Lee, J., Son, M.: Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In: EUROCRYPT. LNCS, vol. 13275, pp. 581–610 (2022)

41. Halevi, S., Shoup, V.: Design and implementation of HElib: a homomorphic encryption library. IACR Cryptol. ePrint Arch. **2020**, 1481 (2020)
42. Hebborn, P., Leander, G.: Dasta - Alternative Linear Layer for Rasta. IACR Trans. Symmetric Cryptol. **2020**(3), 46–86 (2020)
43. Jakobsen, T., Knudsen, L.R.: The Interpolation Attack on Block Ciphers. In: FSE. LNCS, vol. 1267, pp. 28–40 (1997)
44. Kim, S., Ha, J., Son, M., Lee, B., Moon, D., Lee, J., Lee, S., Kwon, J., Cho, J., Yoon, H., Lee, J.: AIM: symmetric primitive for shorter signatures with stronger security. In: CCS. pp. 401–415. ACM (2023)
45. Knudsen, L.R.: Truncated and Higher Order Differentials. In: FSE 1994. LNCS, vol. 1008, pp. 196–211 (1994)
46. Lacan, J., Fimes, J.: Systematic MDS erasure codes based on vandermonde matrices. IEEE Commun. Lett. **8**(9), 570–572 (2004)
47. Lai, X.: Higher Order Derivatives and Differential Cryptanalysis. Communications and Cryptography: Two Sides of One Tapestry (1994)
48. Liu, F., Li, Y., Chen, H., Jiao, L., Luo, M., Wang, M.: YuX: Finite Field Multiplication Based Block Ciphers for Efficient FHE Evaluation. IEEE Transactions on Information Theory pp. 1–1 (2024)
49. Liu, F., Kalam, A., Sarkar, S., Meier, W.: Algebraic Attack on FHE-Friendly Cipher HERA Using Multiple Collisions. IACR Trans. Symmetric Cryptol. **2024**(1), 214–233 (2024)
50. Liu, F., Sarkar, S., Meier, W., Isobe, T.: Algebraic Attacks on Rasta and Dasta Using Low-Degree Equations. In: ASIACRYPT. LNCS, vol. 13090, pp. 214–240 (2021)
51. Liu, F., Sarkar, S., Meier, W., Isobe, T.: The Inverse of χ and Its Applications to Rasta-Like Ciphers. J. Cryptol. **35**(4), 28 (2022)
52. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: EUROCRYPT. LNCS, vol. 765, pp. 386–397 (1993)
53. Méaux, P., Carlet, C., Journault, A., Standaert, F.: Improved Filter Permutators for Efficient FHE: Better Instances and Implementations. In: INDOCRYPT. LNCS, vol. 11898, pp. 68–91 (2019)
54. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: Inscrypt. LNCS, vol. 7537, pp. 57–76. Springer (2011)
55. Naehrig, M., Lauter, K.E., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: CCSW. pp. 113–124. ACM (2011)
56. NIST: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standards and Technology (NIST), FIPS PUB 202, U.S. Department of Commerce (2015)
57. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC. pp. 84–93. ACM (2005)
58. Microsoft SEAL (release 3.7). https://github.com/Microsoft/SEAL (Sep 2021), microsoft Research, Redmond, WA.
59. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. IACR Cryptology ePrint Archive **2007**, 413 (2007)
60. Youssef, A.M., Mister, S., Tavares, S.E.: On the Design of Linear Transformations for Substitution Permutation Encryption Networks. School of Computer Science, Carleton University pp. 40–48 (1997)
61. Youssef, A.M., Mister, S., Tavares, S.E.: On the Design of Linear Transformations for Substitution Permutation Encryption Networks. In: School of Computer Science, Carleton University. pp. 40–48 (1997)

SUPPLEMENTARY MATERIAL

A About Number of Monomials in Pasta$_{\mathtt{v2}}$

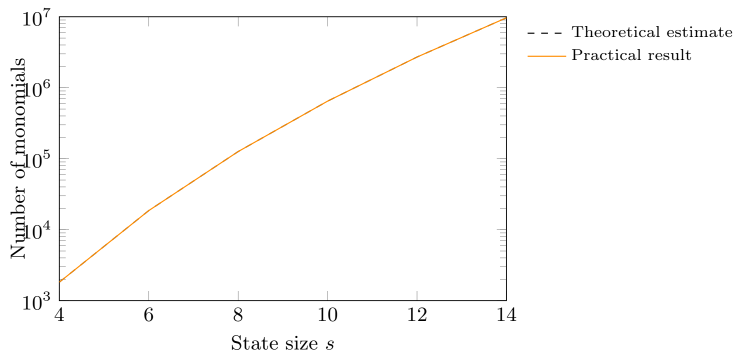


Fig. 5: Estimated number of monomials in each of the output words of Pasta$_{\mathtt{v2}}$ *versus* lowest number of monomials found in a practical evaluation.

B On Truncation

In Pasta$_{\mathtt{v2}}$, we adopt truncation rather than feed-forward operation as in Rasta to compute the keystream words. What if the feed-forward operation is used? In this case, we show an efficient guess-and-determine (GnD) attack. Let us focus on one output of S_{cube} denoted by $x = (x_0, \ldots, x_{2t-1})$. Then, for the feed-forward operation, according to the keystream word W, we have

$$A_r(x) = W - \mathtt{K}\,.$$

Since A_r remains the same for different (N, i), we can always find the inverse of A_r denoted by A_r^{-1} such that

$$x = A_r^{-1}(W - \mathtt{K})\,.$$

Hence, we can guess, say x_0, and compute the corresponding input of this S-box at the last round. This guess can be reused for different (N, i) under the same key $\mathtt{K}$ since A_r^{-1} is fixed and W is known. Hence, the attacker can efficiently peel off the last nonlinear layer by guessing the input of one S-box. Then, it only needs to solve a system of equations in $2t$ variables of degree 2^{r-1}. The total time complexity is $p \cdot \binom{2t+2^{r-1}-1}{2^{r-1}}^{\omega}$ for r round of the new version of Pasta, which also significantly reduces the security of the original Pasta. To prevent this attack, A_r also needs to be randomized, and this contradicts our original goal to minimize the randomness in Rasta-like designs.

C Detailed Proofs – Sect. 7

In this section, we provide the full proofs of the results proposed in Sect. 7.

C.1 Proof of Lemma 1

Denote the i-th row of matrix M as $M_{i,*}$. From the observation $\boldsymbol{e}_i^{(m)} \times M = M_{i,*}$, we obtain

$$\Sigma^{(m)} \times \text{diag}(M)_n = (\text{diag}(M_{0,*})_n, \text{diag}(M_{1,*})_n, \ldots, \text{diag}(M_{m-1,*})_n)^{\mathbf{T}}. \quad (9)$$

Similarly, $N_{i,*}$ denotes the i-th row of matrix N. We have

$$\Sigma^{(n)} \times \text{diag}(N)_m = (\text{diag}(N_{0,*})_m, \text{diag}(N_{1,*})_m, \ldots, \text{diag}(N_{n-1,*})_m)^{\mathbf{T}}. \quad (10)$$

Finally, by multiplying Eq. (9) and Eq. (10), we get matrix Z in Eq. (6).

□

C.2 Proof of Theorem 1 (cont.)

In order to finish the proof started in Sect. 7, we have to consider the cases in which we have more than a single active $\mathbb{F}_q^n$-word in the input. For this goal, we introduce $1 \leq \alpha' \leq m$ as the number of $\mathbb{F}_q^n$-words with at least one active $\mathbb{F}_q$-word:

$$\alpha' = m - \sum_{l=0}^{m-1} \delta_{0,\alpha_l},$$

where $\delta_{l,h}$ is the Kronecker delta (that is, $\delta_{l,h} = 0$ if $l \neq h$, and 1 otherwise).

For each $j \in \{0, 1, \ldots, m-1\}$, let $0 \leq \alpha_j \leq n$ be the number of active $\mathbb{F}_q$-words in $\vec{x}_j$. Working as before, we define β_j as the number of active $\mathbb{F}_q$-words (in the same index set $S_j^{(n)}$) after the application of $\text{diag}(N)_m$. Since N is a MDS matrix:

- $\beta_j = 0$ if and only if $\alpha_j = 0$;
- otherwise, $1 \leq n + 1 - \alpha_j \leq \beta_j \leq n$.

It follows that the number β' of active $\mathbb{F}_q^n$-words does not change, that is, $\beta' = \alpha'$. Moreover, let

$$\beta_{\max} := \max_{j \in \{0,1,\ldots,m-1\}} \beta_j \,,$$

that is, the maximum number of active $\mathbb{F}_q$-word in each active $\mathbb{F}_q^n$-word $\boldsymbol{u}_j$ for $j \in \{0, 1, \ldots, m\}$.

Next, we apply $\Sigma^{(n)}$. Since we are interested in the minimum number of active words in inputs and in outputs, we look for the configuration that minimizes the number of $\mathbb{F}_q^m$-words. By the definition of $\Sigma^{(n)}$, suppose that two input $\mathbb{F}_q$-words $u_{h,i}$ and $u_{h,j}$ are active such that $i, j \in S_h^{(n)} = \{h \cdot n, h \cdot n + 1, \ldots, h \cdot n + n - 1\}$ for a certain $h \in \{0, 1, \ldots, m-1\}$. After applying $\Sigma^{(n)}$, $u_{h,i}$ and $u_{h,j}$ cannot appear in the same $\mathbb{F}_q^m$-word. This is exactly the same as in one active $\mathbb{F}_q^n$-word case. Due to this consideration, after the application of $\Sigma^{(n)}$:

- *at least* $\beta_{\max}$ $\mathbb{F}_q^m$-words are active;
- each active $\mathbb{F}_q^m$-word contains *at most* β' active $\mathbb{F}_q$ words.

After the application of $\text{diag}(M)_n$ where M is an MDS matrix, we have that the number of active words is $1 \leq (m+1-\beta') \cdot \beta_{\max} \leq \gamma \leq n \cdot m$. As a result, the number of active words in inputs and outputs are at least

$$\gamma + \sum_{j=0}^{m-1} \alpha_j \geq (m+1-\beta') \cdot \beta_{\max} + \sum_{j=0}^{m-1} \alpha_j = (m+1-\alpha') \cdot \beta_{\max} + \sum_{j=0}^{m-1} \alpha_j \,.$$

Let's start by considering the simplest case $\beta_{\max} = 1$. In such a case, α_j is either 0 or n for each $j \in \{0, 1, \ldots, m-1\}$ (due to the relation between α_j and β_j, keeping in mind that M is a MDS matrix). In such a case, we have that

$$\gamma + \sum_{j=0}^{m-1} \alpha_j \geq (m+1-\alpha') \cdot \underbrace{\beta_{\max}}_{=1} + \sum_{j=0}^{m-1} \underbrace{\alpha_j}_{\in\{0,n\}} \geq \alpha' \cdot (n-1) + m + 1 \,.$$

By simple computation, $\alpha' \cdot (n-1) + m + 1 < n + m$ (hence, $\alpha' \cdot (n-1) < n-1$) if and only if $\alpha' < 1$, which is not possible since $\alpha' \geq 1$. It follows that the minimum number of active words in inputs and in outputs cannot be smaller than $n + m$.

More generally, if $\beta_{\max} \geq 1$, then $\alpha_j \in \{0, n+1-\beta_{\max}, n+2-\beta_{\max}, \ldots, n\}$ for each $j \in \{0, 1, \ldots, m-1\}$ (due to the relation between α_j and β_j, keeping in mind that M is a MDS matrix). In such a case, we have that

$$\begin{aligned}\gamma + \sum_{j=0}^{m-1} \alpha_j \geq& (m+1-\alpha') \cdot \beta_{\max} + \sum_{j=0}^{m-1} \underbrace{\alpha_j}_{\in\{0,n\}} \\ \geq& (m+1-\alpha') \cdot \beta_{\max} + \alpha' \cdot (n+1-\beta_{\max}) \,.\end{aligned}$$

By simple computation:

$$\begin{aligned}&(m+1-\alpha') \cdot \beta_{\max} + \alpha' \cdot (n+1-\beta_{\max}) < n + m \qquad \leftrightarrow \\ &(m-\alpha') \cdot (\beta_{\max} - 1) + (n - \beta_{\max}) \cdot (\alpha' - 1) < 0 \,.\end{aligned}$$

Note that

- $\alpha' \leq m$ by definition, and that $\beta_{\max} \geq 1$. Hence, the first term is never negative;
- $\beta_{\max} \leq n$ by definition, and that $\alpha' \geq 1$. Hence, the second term is never negative.

It follows that the previous inequality never occurs, which means that the minimum number of active words in inputs and in outputs cannot be smaller than $n + m$. It follows that the branch number is $n + m$.

D Benchmarks (cont.)

In this section, we give the full overview of our benchmark results from Section 6 comparing $\textsc{Pasta}_{\mathsf{v2}}$ and other HHE ciphers in a 17, 33, and 60-bit prime field $\mathbb{F}_p$. First, we provide plain benchmarks in all fields in Table 3. After that, we give the HE benchmarks in the SEAL and HELib homomorphic encryption libraries. Next to the evaluation of homomorphic decompression, we evaluate the use case as proposed in [25] to illustrate the importance of efficient HE parameters. As a use case, we apply three affine layers to a vector of 200 elements (($x'_i = M_i \cdot x_i + b_i$, where $x_i, x'_i, b_i \in \mathbb{F}_p^{200}$, $M_i \in \mathbb{F}_p^{200\times 200}$) interleaved with element-wise squaring on a homomorphically encrypted vector $x \in \mathbb{F}_p^{200}$. This generic use case can be seen as a small 3-layer neural network with squaring activation functions.

HE Parameter Settings. On a high level, we can set three parameters in the BFV and BGV schemes. The polynomial degree $N = \frac{m}{2}$ as a power of two $N = 2^n$, the ciphertext coefficient modulus q, and the plaintext modulus p. The ciphertexts have a noise budget, mostly depleted by multiplications and a security level λ' governed by N and q. When setting parameters, we set p as the minimum modulus feasible as an increasing p adversely affects noise. Increasing q increases our noise budget but diminishes the security parameter, which is compensated by increasing N. Ultimately, increasing the parameters q or N negatively impacts performance. Consequently, minimizing noise expansion during homomorphic decompression is paramount.

D.1 Plain Benchmarks

In Table 3, we present our plain benchmarks in all considered prime fields. The speedup of $\textsc{Pasta}_{\mathsf{v2}}$ compared to pasta is across all primes at least 100%. The rest of the data confirms our points in the main benchmark discussion in Section 6.

D.2 SEAL Benchmarks

First, we discuss the benchmarks in the SEAL library for a 17, 33, and 60-bit prime field $\mathbb{F}_p$. In Table 4, we compare the runtime for homomorphic decompression and the HHE use case when using different instances of $\textsc{Pasta}$ and $\textsc{Pasta}_{\mathsf{v2}}$. In these benchmarks, BFV is parameterized by the degree of the cyclotomic reduction polynomial $N = 2^n$, such that the scheme provides at least 128 bit security and can evaluate the whole circuit without decryption error. One can observe that our changes barely affect the benchmarks in the SEAL library. Runtime differences between $\textsc{Pasta}$ and $\textsc{Pasta}_{\mathsf{v2}}$ are $\approx 1\%$, most likely caused by timing differences from running the benchmarks on a real CPU. Consequently, the additional homomorphic additions and fewer matrix encodings do not significantly affect SEAL and lead to practically equivalent benchmarks when using $\textsc{Pasta}$ or the versions of $\textsc{Pasta}_{\mathsf{v2}}$. In the HHE use case, we sometimes see performance jumps between $\textsc{Pasta}_{\mathsf{v2}}$, $\textsc{Pasta}$, and the remaining ciphers. These substantial differences occur when the lower required noise budget allows for a smaller polynomial degree, drastically impacting performance.

Table 3: Cycles for encrypting one block in plain, averaged over 1000 executions.

Cipher	Total	Instance Generation	Encrypting
$p = 65537$ (17 bit):			
Masta-4	1 970 769	847 971	1 122 798
Masta-5	679 259	328 556	350 703
HERA	**31 874**	**23 536**	**8 338**
Pasta-3	4 054 965	2 473 015	1 581 950
Pasta-4	399 284	293 409	105 875
Pasta$_{v2}$-3	1 950 782	90 735	1 860 047
Pasta$_{v2}$-4	176 999	24 388	152 611
$p = 8088322049$ (33 bit):			
Masta-4	1 862 325	712 804	1 149 521
Masta-5	619 314	262 892	356 422
HERA	**22 294**	**13 607**	**8 687**
Pasta-3	3 978 645	2 312 924	1 665 721
Pasta-4	351 994	250 693	101 301
Pasta$_{v2}$-3	1 956 656	49 645	1 907 011
Pasta$_{v2}$-4	169 632	14 353	155 279
$p = 1096486890805657601$ (60 bit):			
Masta-4	2 317 877	704 746	1 613 131
Masta-5	755 497	258 697	496 800
HERA	**21 691**	**13 305**	**8 386**
Pasta-3	5 376 040	2 966 305	2 409 735
Pasta-4	457 585	295 013	162 572
Pasta$_{v2}$-3	2 696 096	49 546	2 646 550
Pasta$_{v2}$-4	227 427	13 967	213 460

D.3 HElib Benchmarks

In Table 5, we compare the runtime for homomorphic decompression and the HHE use case when using different instances of Pasta and Pasta$_{v2}$. Additionally to the data in Section 6, we display the selected modulus degree $m = 2 \cdot N$ and the computed HE security parameter λ'. Several things can be seen in this table. First, the performance benefit of Pasta$_{v2}$ spans across all plaintext parameters. Second, the increased noise of the HElib rotation implementation further emphasizes the relevance of a low multiplicative depth. Finally, the Masta-5 and HERA use cases in the 33-bit prime setting, the basic HERA and Masta-5 decompression, and all the HHE use case evaluations in the 60-bit prime setting yielded insecure parameters. Increasing m further would lead to polynomials with 131 072 coefficients. Necessary RNS decomposition of the large ciphertext modulus then leads to large encryptions of the secret key and infeasible memory consumption on the server side.

Table 4: Benchmarks for the SEAL library.

Cipher	1 Block			HHE use case			
	N	Enc. Key	Decomp.	N	Enc. Key	Decomp.	Use Case
		s	s		s	s	s
$p = 65537$ (17 bit):							
Pasta-3	16384	0.008	4.44	32768	0.031	43.4	22.1
Pasta-4	16384	0.008	2.02	32768	0.029	69.9	20.9
Masta-4	16384	0.008	5.59	32768	0.029	51.8	20.9
Masta-5	32768	0.033	19.7	32768	0.028	74.3	20.9
HERA	32768	0.030	8.20	32768	0.028	105.6	20.8
Pasta$_{v2}$-3	16384	0.008	4.63	32768	0.031	43.1	22.1
Pasta$_{v2}$-4	16384	0.008	2.12	32768	0.031	73.9	22.3
$p = 8088322049$ (33 bit):							
Pasta-3	32768	0.033	21.8	32768	0.029	43.4	22.2
Pasta-4	32768	0.031	10.1	65536	0.118	414.1	109.8
Masta-4	32768	0.031	25.9	65536	0.111	273.1	109.7
Masta-5	32768	0.028	19.1	65536	0.111	406.2	110.4
HERA	32768	0.026	8.16	65536	0.112	592.6	105.6
Pasta$_{v2}$-3	32768	0.032	21.6	32768	0.028	43.0	22.0
Pasta$_{v2}$-4	32768	0.032	10.6	65536	0.117	410.5	109.4
$p = 1096486890805657601$ (60 bit):							
Pasta-3	32768	0.029	29.2	65536	0.125	223.0	109.8
Pasta-4	65536	0.118	56.0	65536	0.112	414.7	110.0
Masta-4	65536	0.118	132.6	65536	0.111	272.5	103.0
Masta-5	65536	0.118	99.5	65536	0.111	423.4	109.9
HERA	65536	0.119	46.4	65536	0.112	610.4	109.9
Pasta$_{v2}$-3	32768	0.028	28.9	65536	0.124	221.4	109.4
Pasta$_{v2}$-4	65536	0.125	58.9	65536	0.111	411.3	109.4

Table 5: Benchmarks for the HElib library.

Cipher	m	λ' bit	Enc. Key s	Decomp. s	m	λ' bit	Enc. Key s	Decomp. s	Use Case s
$p = 65537$ (17 bit):									
Pasta-3	65536	184	0.033	14.7	65536	128	0.035	33.8	11.5
Pasta-4	65536	163	0.033	6.93	131072	229	0.065	116.0	21.4
Masta-4	65536	163	0.038	20.0	131072	229	0.063	84.2	24.6
Masta-5	65536	133	0.045	16.5	131072	199	0.073	140.8	28.0
HERA	131072	254	0.071	11.5	131072	189	0.072	178.4	28.0
Pasta$_{v2}$-3	65536	184	0.031	12.3	65536	128	0.040	27.0	11.6
Pasta$_{v2}$-4	65536	163	0.033	5.95	131072	229	0.068	102.1	21.5
$p = 8088322049$ (33 bit):									
Pasta-3	65536	125	0.037	17.7	131072	162	0.106	112.3	38.1
Pasta-4	131072	204	0.099	21.4	131072	144	0.112	182.7	34.3
Masta-4	131072	196	0.095	57.3	131072	144	0.101	150.4	40.7
Masta-5	131072	166	0.096	49.8	131072[a]	117	0.131	250.9	45.4
HERA	131072	150	0.108	17.2	131072[a]	110	0.145	307.6	51.8
Pasta$_{v2}$-3	65536	125	0.033	15.0	131072	162	0.099	97.2	36.5
Pasta$_{v2}$-4	131072	204	0.094	18.2	131072	144	0.118	163.8	34.9
$p = 1096486890805657601$ (60 bit):									
Pasta-3	131072	162	0.118	57.4	131072[a]	97	0.151	162.6	51.0
Pasta-4	131072	129	0.130	29.3	131072[a]	83	0.167	276.3	50.1
Masta-4	131072	129	0.107	84.3	131072[a]	83	0.161	217.3	56.8
Masta-5	131072[a]	99	0.132	71.2	131072[a]	70	0.186	354.0	64.6
HERA	131072[a]	89	0.147	26.4	131072[a]	60	0.200	466.7	75.5
Pasta$_{v2}$-3	131072	162	0.113	48.9	131072[a]	97	0.151	138.9	45.7
Pasta$_{v2}$-4	131072	129	0.116	26.9	131072[a]	83	0.165	251.1	50.1

[a] Further increasing m for security resulted in infeasibly long runtimes.