From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications

Lorenzo Grassi[1], Morten Øygarden[2], Markus Schofnegger[3], and Roman Walch[3,4]

[1] Radboud University Nijmegen (The Netherlands)
[2] Simula UiB (Norway)
[3] Graz University of Technology (Austria)
[4] Know-Center GmbH (Austria)
`lgrassi@science.ru.nl`
`morten.oygarden@simula.no`
`firstname.lastname@iaik.tugraz.at`

Abstract. Ciminion is an MPC-friendly pseudo-random function (PRF) recently proposed at Eurocrypt'21. As in the case of other MPC-friendly constructions proposed in the literature (e.g., MiMC, HadesMiMC, *Rescue*), it aims to minimize the number of multiplications in large finite fields. While MiMC, HadesMiMC, and *Rescue* are block ciphers, Ciminion is a (modified) Farfalle-like cryptographic function. At the current state of the art, it achieves the best performance in MPC applications. However, Ciminion has a critical weakness. Its security highly relies on the independence of its subkeys, which is achieved by using an expensive key schedule. Many MPC use cases involving symmetric PRFs rely on secretly shared symmetric keys, and hence the expensive key schedule must also be computed in MPC. As a result, Ciminion's performance is significantly reduced in these use cases.

In this paper, we solve this problem. Following the approach introduced by Ciminion's designers, we propose Megafono, a modified version of Farfalle designed for achieving a small multiplicative complexity without any key schedule. Following this strategy, we present the PRF Hydra, which utilizes both a Lai–Massey construction and a novel construction we name `Amaryllises` in its nonlinear layer. `Amaryllises` can be seen as a generalized variant of a Lai–Massey scheme, which allows us to further decrease the multiplicative complexity of Hydra.

Based on an extensive security analysis, we implement Hydra in an MPC framework. The results show that it outperforms all MPC-friendly schemes currently published in the literature.

Keywords: Megafono – Hydra – Farfalle – Ciminion – Lai–Massey – `Amaryllises` – MPC Applications

1 Introduction

Secure multi-party computation (MPC) allows several parties to jointly and securely compute a function on their combined private inputs. Thereby, the correct output is computed and given to all (or a subset of the) parties while simultaneously hiding the private inputs from other parties. In this work we focus on secret-sharing based MPC schemes, such as the popular SPDZ protocol [22, 21], or protocols based on Shamir's Secret Sharing [51]. In these protocols private data is shared among all parties, such that each party receives a share which – on its own – does not contain any information about the initial data. When combined, however, the parties are able to reproduce the shared value. Furthermore, the parties can use these shares to compute complex functions on the data which in turn produce shares of the output.

In recent years, MPC has been applied to many use cases, including privacy-preserving machine learning [49], private set intersection [39], truthful auctions [14], and revocation in credential systems [36]. In the literature describing these use cases, data is often directly entered from and delivered to the respective parties. However, in practice, this data often has to be transferred securely from and to third parties before it can be used in the MPC protocol. Besides that, in some applications, intermediate results of an MPC computation may need to be stored securely in a database. As described in [31], one can use MPC-friendly Pseudo-Random Functions (PRFs), i.e., PRFs designed to be efficient in MPC applications, to efficiently realize this secure data storage and data transfer by directly encrypting the data using a secret-shared symmetric key.

These MPC-friendly PRFs, however, cannot only be used to securely transmit data in given MPC computations, they can also be used as a building block to speed up many MPC applications, such as secure database join via an MPC-evaluation of a PRF [44], distributed data storage [31], virtual hardware security modules[5], MPC-in-the-head based zero-knowledge proofs [37] and signatures [16], oblivious TLS [1], among many others. *In all these use cases, the symmetric encryption key is shared among all participating parties. Consequently, if one has to apply a key schedule for a given PRF, one has to compute this key schedule at least once in MPC for every fresh symmetric key.*

In order to be MPC-friendly, a PRF must minimize the number of multiplications in the native field of the MPC protocol. At the current state of the art, Ciminion [24] is one of the most competitive symmetric encryption schemes for PRF applications. Proposed at Eurocrypt'21, it is based on the Farfalle mode of operation [9]. However, as we will show in detail, Ciminion has a serious drawback: *Its security heavily relies on the assumption that the subkeys are independent.* In order to satisfy this requirement, the subkeys are generated via a sponge hash function [10] instantiated with an expensive permutation from the point of view of the multiplicative complexity. As a result, in all the (common) cases in which the key is shared among the parties, the key schedule cannot be computed locally and also needs to be evaluated in MPC. This leads to a significant increase in the multiplicative complexity of Ciminion. In this paper, we handle this problem in two steps. First, we propose a new mode of operation inspired by Farfalle and Ciminion, called Megafono, which is designed

[5] https://www.unboundsecurity.com/wp-content/uploads/2020/09/vHSM-3.pdf

to be competitive in all MPC applications.[6] Secondly, we show how to instantiate it in order to minimize the multiplicative complexity. The obtained PRF called Hydra is the most competitive MPC-friendly PRF currently present in the literature.

1.1 Related Works: Ciminion and the MPC Protocols

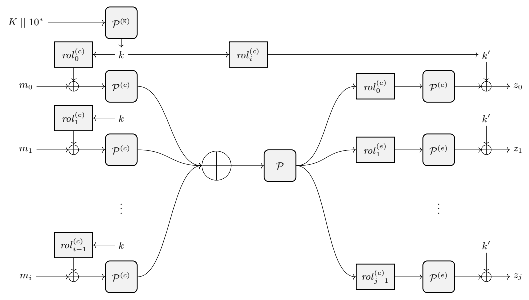
Modern MPC protocols are usually split into a data-independent *offline phase* and a data-dependent *online phase*. In the offline phase, a bundle of shared correlated randomness, most notably beaver triples [8], is generated, which is then used in the online phase to perform the actual computation on the private data. The performance scales with the number of nonlinear operations. Indeed, each multiplication requires one beaver triple, which is computed in the offline phase, as well as one round of communication during the online phase. In contrast, linear operations do not require any offline computations and can directly be applied to the shares without communication. Consequently, MPC-friendly PRFs usually try to minimize the number of multiplications.

Traditional PRFs such as AES or Keccak/SHA-3 are not efficient in MPC settings. First, MPC applications usually work over a prime field $\mathbb{F}_p$ for a large p (e.g., $p \approx 2^{128}$), while traditional cryptographic schemes are usually bit-/byte-oriented schemes. Hence, a conversion from $\mathbb{F}_{2^n}$ to $\mathbb{F}_p$ and vice versa must take place, which can impact the overall performance. Secondly, and most importantly, traditional schemes are designed to minimize their implementation cost in software and/or hardware, and therefore no particular focus is laid on minimizing specifically the number of nonlinear operations (e.g., `AND` gates).

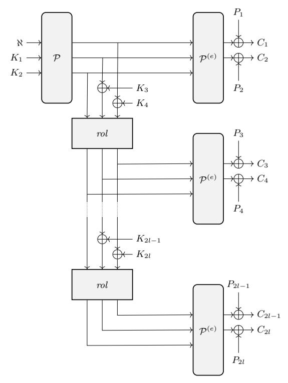
For these reasons, several MPC-friendly schemes over $\mathbb{F}_q^t$ for $q = p^s$ and $t \geq 1$ have been proposed in the literature, including LowMC [4], MiMC [3], GMiMC [2], HadesMiMC [28], and *Rescue* [5]. All those schemes are block ciphers – hence, invertible – and they are often used in counter (CTR) mode. However, the invertibility property is not required in MPC applications, and a lower multiplicative complexity can (potentially) be achieved by working with non-invertible functions, as recently shown by Dobraunig et al. [24]. In the following, we briefly discuss the Farfalle construction and the MPC-friendly primitive Ciminion based on it.

Farfalle. Farfalle [9] is an efficiently parallelizable permutation-based construction of arbitrary input and output length, taking as input a key. As shown in Fig. 1a and recalled in Section 2.1, the Farfalle construction consists of a *compression layer* followed by an *expansion layer*. The compression layer produces a single accumulator value from the input data. A permutation is potentially applied to the obtained state. Then, the expansion layer transforms it into a

[6] "Megafono" is the Italian word for "megaphone", a cone-shaped horn used to amplify a sound and direct it towards a given direction. Our strategy resembles this goal.



(a) The Farfalle construction.



(b) The Ciminion PRF.

Fig. 1: Farfalle and Ciminion (notation adapted to the one used in this paper).

tuple of (truncated) output blocks. Both the compression and expansion layers involve the secret key, and they are instantiated via a set of permutations (namely, $\mathcal{P}^{(c)}, \mathcal{P}, \mathcal{P}^{(e)}$) and rolling functions.

Since an attacker never has access to both the input and output of a permutation call, the number of rounds of the involved iterated permutations is, in general, *smaller* than in e.g. sponge/duplex constructions [10] and/or block ciphers (used e.g. in CTR mode).

Ciminion. At Eurocrypt'21, Dobraunig et al. [24] showed that a modified version of Farfalle can be competitive for MPC protocols, an application which the Farfalle's designers did not take into account. They propose the PRF Ciminion, a modified version of Farfalle instantiated with a Feistel scheme, whose round function is defined via the Toffoli gate $(x, y) \mapsto x \cdot y$. As shown in Fig. 1b, and recalled in Section 2.2

(1) compared to Farfalle, the compression phase is missing, a final truncation is applied, and the key addition is performed before $\mathcal{P}^{(e)}$ is applied, and
(2) in contrast to MPC-friendly block ciphers, Ciminion is a *non-invertible* PRF. For encryption it is used as a stream cipher, where the input is defined as the concatenation of the secret key and a nonce.

The main reason why Ciminion is currently the most competitive scheme in MPC protocols is related to one crucial feature of Farfalle, that is, the possibility to instantiate its internal permutations with a smaller number of rounds compared to other design strategies. In the case of Ciminion, the permutation $\mathcal{P}^{(c)}$ is designed in order to behave like a pseudo-random permutation, while the number of rounds of the permutation $\mathcal{P}^{(e)}$ is kept as low as possible in order to provide security and achieve good performance.

Besides minimizing the number of nonlinear operations, Ciminion's designers paid particular attention to the number of linear operations. Indeed, even though the main cost in MPC applications depends on the number of multiplications, other factors (e.g., the number of linear operations) affect efficiency as well.

1.2 The Megafono Design Strategy

The main drawback of Ciminion is the expensive key schedule to generate subkeys that can be considered independent. This implies that Ciminion only excels in MPC applications where the key schedule can be precomputed for a given shared key, or in the (non-common) scenarios where the key is not shared among the parties. However, in the latter case the party knowing the key can also compute Ciminion's keystream directly in plain (i.e., without MPC) if the nonce and IV are public in a given use case (which is also true for any stream cipher).

Clearly, the easiest possible solution is the removal of the nonlinear key schedule. However, by e.g. defining the subkey as an affine transformation of the master key, the security analysis of Ciminion does not hold anymore.

As we are going to discuss in detail in Section 3, the problem follows from the Farfalle construction itself. Even if the attacker does not have any information about the internal states of Farfalle, they can exploit the fact that its outputs are generated from the **same** unknown input (namely, the output of $\mathcal{P}^{(c)}$ and/or $\mathcal{P}$). Given these outputs and by exploiting the relations of the corresponding unknown inputs (which are related to the definition of the rolling function), the attacker can potentially find the key and break the scheme. For example, the attacks [15, 18] on the Farfalle schemes Kravatte and Xoofff exploit exactly this strategy. In Ciminion, this problem is solved by including an addition with the secret master key in the application of the rolling function. In this way, the mentioned relation is unknown due to the presence of the key, and $\mathcal{P}^{(e)}$ can be instantiated via an efficient permutation.

We make two crucial changes in the Farfalle design strategy. We first replace the permutation $\mathcal{P}^{(e)}$ with a keyed permutation $\mathcal{P}_{\mathtt{K}}^{(e)}$, where a key addition takes place in each round. Secondly, we expand the input of this keyed permutation. The second change aims to frustrate algebraic attacks, whose cost is related to both the degree and the number of variables of the nonlinear equation system representing the attacked scheme. In order to create new independent variables for "free" (i.e., without increasing the overall multiplicative complexity), we reuse the computations needed to evaluate $\mathcal{P}$. That is, we define the new variable as the sum of all the internal state of $\mathcal{P}$, and we conjecture that it is sufficiently independent of its output (details are provided in the following).

Finally, we replace the truncation in Ciminion with the summation-truncation hybrid technique introduced in [33], in order to reduce the amount of the discarded randomness without any impact on the security. Our result is a new design strategy which we call Megafono.

1.3 The PRF Hydra

Given the mode of operation, we instantiate it with two distinct permutations, one for the initial phase and one for the expansion phase. As in Ciminion, assuming the first permutation behaves like a pseudo-random permutation and since the attacker does not know the internal states of Megafono, it is possible to guarantee security of the overall design by choosing a second permutation that is significantly cheaper to evaluate in the MPC setting than the first one. In particular, it is crucial to keep in mind that while the first permutation is evaluated only once, the number of calls to the second permutation (and so the overall cost) is proportional to the output size.

In order to minimize the multiplicative complexity, we focused on quadratic functions for the definition of the round functions of the second permutation. However, since no quadratic function is invertible over $\mathbb{F}_p$, we use them in a mode of operation that guarantees invertibility. We opted for the generalized Lai–Massey constructions recently proposed in [29].

While the round function of the second (keyed) permutation is instantiated with a quadratic Lai–Massey scheme, the first permutation uses the Hades design strategy [28]. Its main feature regards the use of both rounds with full S-box layers and rounds with partial S-box layers in order to achieve both security and good performance. Following Neptune [29], we use two different round functions, one for the internal part and one for the external one. We decided to instantiate the internal rounds with a Lai–Massey scheme. For the external rounds, we modified the original Lai–Massey scheme $(x_0, x_1, \ldots, x_{t-1}) \mapsto (x_0 + F(\sum_j \lambda_j \cdot x_j), x_1 + F(\sum_j \lambda_j \cdot x_j), \ldots, x_{t-1} + F(\sum_j \lambda_j \cdot x_j))$, where $\sum_j \lambda_j = 0$ by modifying the sum with the multiplication operation, that is, $(x_0, x_1, \ldots, x_{t-1}) \mapsto (x_0 \times F'(\sum_j \lambda'_j \cdot x_j), x_1 + F'(\sum_j \lambda'_j \cdot x_j), \ldots, x_{t-1} + F'(\sum_j \lambda'_j \cdot x_j))$ for a function F', with specific properties given in Theorem 1. In Section 5, we show how to instantiate F' in order to guarantee invertibility and minimize the number of multiplications. *This modified Lai–Massey construction, which we call* ***Amaryllises***,[7] *ensures to mix the state in a nonlinear way.*

The obtained PRF scheme called Hydra is presented in Section 4 – Section 6, and its security analysis is proposed in Section 7 – Section 8.

1.4 MPC Performance and Comparison

As discussed above, the performance of any MPC calculation scales with the number of nonlinear operations, in other words, multiplications of shared elements. In Figure 2 we compare the number of these multiplications which are required to evaluate different PRFs for different state sizes t using secret shared keys. One can observe that Hydra requires the smallest number of multiplication, with the difference growing further for larger state sizes. The only PRF that is competitive to Hydra is Ciminion, and only *if the key schedule does not have to be computed* which is only the case if shared round keys can be reused from a previous computation. However, this implies that the key schedule was already computed once in MPC, requiring a significant amount of multiplications. Hydra, on the other hand, does not require the computation of an expensive

[7] The flowers "ama(r)yl(l)ises" is (almost) the anagram of Lai–Massey.

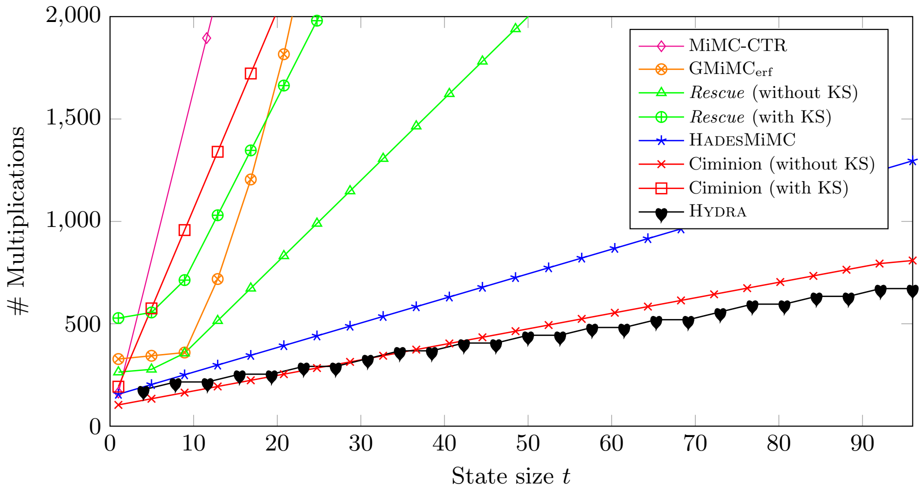


Fig. 2: Number of MPC multiplications of several designs over $\mathbb{F}_p^t$, with $p \approx 2^{128}$ and $t \geq 2$ (security level of 128 bits).

key schedule and also requires fewer multiplications than Ciminion without a key schedule for larger state sizes.

In Section 9, we implement and compare the different PRFs in the MP-SPDZ [40] library and confirm the performance expected from Figure 2. Concretely, taking key schedules into account, Hydra is 5 times faster than Ciminion for $t = 8$, which grows to a factor of 19 for $t = 128$. Without key schedules, Ciminion is only slightly faster than Hydra for smaller t, until it gets surpassed by Hydra for $t > 64$, showing that Hydra is also competitive, even if the round keys are already present. Compared to all other benchmarked PRFs, Hydra is significantly faster for any state size t. Furthermore, Hydra requires the least amount of communication between the parties due to its small number of multiplications, giving it an advantage in low-bandwidth networks.

2 Starting Points: Farfalle and Ciminion

Notation. We work over a finite field $\mathbb{F}_q$, where $q = p^s$, for an odd prime number p and $s \geq 1$ an integer (when needed, we will also assume a fixed vector space isomorphism $\mathbb{F}_{p^s} \equiv \mathbb{F}_p^s$). The $\cdot \mid\mid \cdot$ operator denotes the concatenation of two elements. Given $x \in \mathbb{F}_q^r$, it is represented as $x = (x_0, x_1, \ldots, x_{r-1})$, where x_i denotes its i-th element. We use the notation $\mathbb{F}_q^\star$ to denote $\mathbb{F}_q$ strings of arbitrary length. We use the fraktur font notation to denote a subspace of $\mathbb{F}_q^r$, while we sometimes use the calligraphic notation to emphasize functions.

Definition 1. *Let $a, b \in \mathbb{N} \setminus \{0\}$ be such that $a > b$. We define respectively the left and the right truncation functions $\mathcal{T}_{a,b}^L : \mathbb{F}_q^a \to \mathbb{F}_q^b$ and $\mathcal{T}_{a,b}^R : \mathbb{F}_q^a \to \mathbb{F}_q^b$ as*

$$\mathcal{T}_{a,b}^L(x_0, \ldots, x_{a-1}) = (x_0, \ldots, x_{b-1}), \quad \mathcal{T}_{a,b}^R(x_0, \ldots, x_{a-1}) = (x_{a-b-1}, x_{a-b}, \ldots, x_{a-1}).$$

Clearly, for each $x \in \mathbb{F}_q^t$ and for each $i \in \{1, \ldots, t-2\}$, $x = \mathcal{T}^L_{t,i}(x) \,||\, \mathcal{T}^R_{t,t-i}(x)$.

2.1 Farfalle Strategy and $^{1/2\times}$Farfalle

Farfalle is a keyed pseudo-random function proposed in [9]. Both its input and its output are of an arbitrary length. As shown in Fig. 1a, it has a compression layer and an expansion layer, each involving the parallel application of a permutation. We refer to Appendix B for a generic description of Farfalle.

For the goal of this paper, we focus only on the expansion phase, and we use the term $^{1/2\times}$Farfalle for identifying the specific version of Farfalle in which the initial compression phase is missing, or equivalently, if the input message is an element of $\mathbb{F}_q$. Let $\mathtt{K} \in \mathbb{F}_q^\kappa$ be the secret key for $\kappa \geq 1$. $^{1/2\times}$Farfalle uses

(1) a key schedule $\mathcal{K} : \mathbb{F}_q^\kappa \to \mathbb{F}_q^\star$ for generating the subkeys used in the expansion phase,[8] that is, $\mathcal{K}(\mathtt{K}) = (k_0, k_1, \ldots, k_n, \ldots)$,
(2) two unkeyed functions $\mathcal{F}, \mathcal{F}^{(e)} : \mathbb{F}_q \to \mathbb{F}_q$, and
(3) an expansion function $\mathcal{E} : \mathbb{F}_q \to \mathbb{F}_q^\star$ defined as $\mathcal{E}(x) = (\mathcal{E}_0(x), \mathcal{E}_1(x), \ldots)$ for $\mathcal{E}_i : \mathbb{F}_q \to \mathbb{F}_q$. Given $\mathcal{E}' : \mathbb{F}_q \to \mathbb{F}_q$, called the *rolling function*, $\mathcal{E}_i$ can be defined iteratively as $\mathcal{E}_i(x) = \mathcal{E}' \circ \mathcal{E}_{i-1}(x)$ for a certain function $\mathcal{E}_0$ (including e.g. $\mathcal{E}_0(x) = \mathcal{E}'(x)$ or e.g. the identity function $\mathcal{E}_0(x) = x$).

Given an input $x \in \mathbb{F}_q$, $^{1/2\times}\text{Farfalle} : \mathbb{F}_q \to \mathbb{F}_q^n$ operates as

$$^{1/2\times}\text{Farfalle}(x) = (\underbrace{k_1 + \mathcal{F}^{(e)}, \ldots, k_n + \mathcal{F}^{(e)}}_{n \text{ times}}) \circ \mathcal{E} \circ \mathcal{F}(x + k_0),$$

where $(k_1 + \mathcal{F}^{(e)}, \ldots, k_n + \mathcal{F}^{(e)})(z) \coloneqq (k_1 + \mathcal{F}^{(e)}(z_0), \ldots, k_n + \mathcal{F}^{(e)}(z_{n-1}))$ given $z = (z_0, \ldots, z_{n-1}) \in \mathbb{F}_q^n$.

2.2 From $^{1/2\times}$Farfalle to Ciminion

Ciminion [24] is based on a modified version of the $^{1/2\times}$Farfalle design strategy over $\mathbb{F}_q \equiv \mathbb{F}_{p^s}$ for $s \geq 2$. As shown in Fig. 1b, the main difference with respect to $^{1/2\times}$Farfalle regards the definition of the function $k + \mathcal{F}^{(e)}$. In Farfalle/$^{1/2\times}$Farfalle, the function $\mathcal{F}^{(e)}$ is usually instantiated via a permutation $\mathcal{P}^{(e)}$ over $\mathbb{F}_q$, and the key addition is the last operation to take place. In Ciminion, $k + \mathcal{P}^{(e)}(\cdot)$ is replaced by $\mathcal{F}^{(e)}(\cdot + k) \coloneqq \mathcal{T}^L_{s,s'} \circ \mathcal{P}^{(e)}(\cdot + k)$ for a *non-invertible* function $\mathcal{F}^{(e)}$ instantiated via a truncated permutation. Moving the key inside the scheme allows to prevent its cancellation when considering the difference of two outputs.

In Ciminion, the key schedule $\mathcal{K} : \mathbb{F}_q \to \mathbb{F}_q^\star$ uses a sponge hash function [10], instantiated via the permutation $\mathcal{P}$. We refer to [24, Section 2] for more details.

[8] We mention that in [9], authors use the terms "masks" and "rolling function" instead of "subkeys" and "key schedule". In Farfalle, the same subkey is used for in the expansion phase, that is, $k_1 = k_2 = \cdots = k_n = \cdots$. Here, we consider the most generic case in which the subkeys are not assumed to be equal.

3 The Megafono Strategy for Hydra

The security argument proposed by the designers of Ciminion heavily relies on the subkeys $k_0, k_1, \ldots$ being linearly independent. To achieve this goal, such subkeys are generated via a sponge hash function built from a permutation $\mathcal{P}$ that resembles a Pseudo-Random Permutation (PRP). This approach is expensive in terms of multiplications needed. As a result, at the current state, Ciminion is the most competitive scheme *only in the case in which the key schedule does not have to be computed.* However, if the secret keys are shared among the parties, the key schedule has to be computed in MPC for every new key, and Ciminion is not competitive against other MPC-friendly schemes such as e.g. HadesMiMC.

Another (small) weakness of Ciminion regards the final truncation. It is crucial for the security, since without it one could easily compute the inverse of the final permutations $\mathcal{P}^{(e)}$ and potentially break the scheme. However, the truncation is wasteful, in the sense that one can achieve the same result of preventing the inverse computation of $\mathcal{P}^{(e)}$ by e.g. using a different approach as the feed-forward one or by replacing the permutation with a non-invertible function.

With Hydra, our goal is to propose an MPC-friendly scheme that takes into account the advantages of Ciminion and at the same time does not have its limitations. In particular, our aim is to design an MPC-friendly PRF that is competitive for both cases in which a fresh key is shared among the parties and an old one is reused. As we are going to show, reaching this goal requires crucial modifications in the design strategy of Ciminion (and so of $^{1/2\times}$Farfalle). We refer to the design strategy that we are going to present as the Megafono design strategy.

3.1 Initial Considerations

The Summation-Truncation Hybrid. In Ciminion, the final truncation on each permutation call is crucial in order to prevent backward attacks. However, outright truncation is wasteful, in the sense that one can make more economical use of the discarded randomness without any impact on the security. A way to achieve this was demonstrated at Crypto 2020 in [33], where authors present the summation-truncation technique. In such a case, the idea is to combine two well known techniques used to set up a Pseudo-Random Function (PRF) given Pseudo-Random Permutations (PRPs), i.e., the truncation one [35] and the sum of two independent permutations [47]. Let $1 \le s' < s$. Given two (keyed) permutations $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}$ over $\mathbb{F}_{p^s}$, let $\mathcal{F} : \mathbb{F}_{p^s} \to \mathbb{F}_{p^{s+s'}}$ be defined as

$$\mathcal{F}(x) = \mathcal{T}^L_{s,s'} \circ \mathcal{P}^{(1)}(x) \middle\| \mathcal{T}^R_{s,s-s'} \circ \mathcal{P}^{(1)}(x) + \mathcal{T}^L_{s,s-s'} \circ \mathcal{P}^{(2)}(x) \middle\| \mathcal{T}^R_{s,s'} \circ \mathcal{P}^{(2)}(x).$$

The resulting summation-truncation function has the *same* security level of a truncation function (that is, it is indistinguishable from a PRF up to $p^{s-s'/2}$ queries), while the size of the output is obviously bigger.

Based on this simple consideration, we decided to replace the final truncation of Ciminion with the summation-truncation one. For the follow-up, we recall that two independent permutations can be set up via a single permutation $\mathcal{P}$ through domain separation by fixing part of the inputs [47]. That is, given a PRP $\mathcal{P}$, let $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ be defined as $\mathcal{P}^{(j)}(x) \coloneqq \mathcal{P}(x\|i_j)$ for $j \in \{1,2\}$, with the only condition that $i_1 \neq i_2$. Then, $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ are two independent PRPs.

Removing the Key Schedule. As we mentioned before, in order to make the permutation $\mathcal{P}^{(e)}$ as cheaper as possible, the designers of Ciminion assume the subkeys $k_0, k_1, k_2, \ldots$ to be independent. This result can be achieved by working with a key schedule $\mathcal{K}$ that simulates a PRF, *expensive* in term of multiplicative complexity. In order to fix this, one option would consist in considering a lighter (but still nonlinear) key schedule. In such a case, one should adapt the security analysis of Ciminion to the case in which the subkeys are not independent. Instead, we decided to *remove the key schedule completely.*

3.2 From $^{1/2\times}$Farfalle to the Megafono Design Strategy

Instead of relying on independent subkeys for security, we work as follows.

– We define each subkey to be equal to the master key $\mathtt{K}$ (or an affine transformation of it).
– We perform a key addition *before* the expansion phase.
– Instead of having independent subkeys, we create new $\mathbb{F}_q$ elements at the input of $\mathcal{P}^{(e)}$ by reusing the operations necessary for computing $\mathcal{P}_{\mathtt{K}}$.
– We replace the (unkeyed) permutation in the expansion phase with a keyed permutation in which a key addition takes place in each round.

Due to the last point, we will use the notation $\mathcal{P}_{\mathtt{K}}^{(e)}$ to emphasize the presence of round key additions.

Key Addition Before the Expansion Phase. In Ciminion, the permutation $\mathcal{P}$ is chosen in order to resemble a PRP. By performing a key addition before the expansion phase, the first part of the scheme becomes an Even–Mansour construction [26] of the form

$$x \mapsto \mathcal{P}_{\mathtt{K}}(x) \coloneqq \mathtt{K} + \mathcal{P}(x + \mathtt{K}). \tag{1}$$

Assuming $\mathcal{P}_{\mathtt{K}}$ is a PRP and if the attacker knows both the inputs and the outputs, the security of this part is equal to $q^{t/2}$ for $\mathtt{K} \in \mathbb{F}_q^t$, as proven in [19, 25]. This allows us to make a security claim on a subcomponent of the entire scheme, and so to simplify the overall security analysis.

Keyed Permutation. In Ciminion, the key has been moved from the end of the permutation $\mathcal{P}^{(e)}$ to the beginning in order to avoid its cancellation by considering differences of the outputs. At the same time, by considering differences of the (unknown) inputs, it is possible to eliminate the key as a variable in the system of equations that describe the expansion phase. In order to avoid this problem, we decided to consider a keyed permutation $\mathcal{P}_{\mathtt{K}}^{(e)}$, that is, a permutation in which the key addition takes place at each round.

Creating New Variables. Due to the structure of ${}^{1/2\times}$Farfalle and of Ciminion, and under the assumption in which $\mathcal{P}$ behaves like a PRP, an attacker cannot control the inputs and the outputs of the expansion phase. At the same time, attacks that require only the knowledge of the outputs of such and expansion phase are possible, due to the fact that multiple outputs are created via a single *common* (unknown) input. E.g., several attacks on the expansion phase of Farfalle schemes are based on the Meet-in-the-Middle approach [15, 18], that is, the attacker exploits the relation among several inputs of $\mathcal{P}^{(e)}$ for breaking the scheme. In such a scenario, another powerful attack is the Gröbner basis one. As we are going to explain in more details in Section 8.2, Gröbner basis is a technique that allows to solve/factorize a system of nonlinear equations. The cost of such attack is proportional to the number of unknown and independent variables and on the degree of the equation to solve. Hence, in order to guarantee security against such attack, it is necessary to increase the degree of the equations or/and the number of unknown and independent variables. In Ciminion, this goal is achieved by introducing new independent sub-keys $k_0, k_1, \ldots$ at the input of every $\mathcal{P}^{(e)}$ call, where the subkeys are generated by a pseudo-random function. However, as we have already pointed out, this turns out not to be efficient when the key is shared among the parties.

In order to increase the number of variables "for free", we propose to re-use the computation needed to evaluate $\mathcal{P}_{\mathtt{K}}$. Since $\mathcal{P}_{\mathtt{K}}$ will be instantiated as an iterated permutation, we can fabricate a new $\mathbb{F}_q$ element by considering e.g. the sum of all internal states of $\mathcal{P}_{\mathtt{K}}$. More formally, let

$$\mathcal{P}_{\mathtt{K}}(x) \coloneqq \mathtt{K} + \mathcal{P}_{r-1} \circ \mathcal{P}_{r-2} \circ \cdots \circ \mathcal{P}_1 \circ \mathcal{P}_0(\mathtt{K} + x) \tag{2}$$

for some permutations $\mathcal{P}_{r-1}, \mathcal{P}_{r-2}, \ldots, \mathcal{P}_1, \mathcal{P}_0 : \mathbb{F}_q \to \mathbb{F}_q$ and for $r \gg 1$ such that $\mathcal{P}_{\mathtt{K}}$ resembles a PRP. For a given input $x \in \mathbb{F}_q$, let $y, z \in \mathbb{F}_q$ be defined as

$$y \coloneqq \mathcal{P}_{\mathtt{K}}(x), \qquad z \coloneqq \sum_{i=0}^{r-2} \mathcal{P}_i \circ \mathcal{P}_{i-1} \circ \cdots \circ \mathcal{P}_1 \circ \mathcal{P}_0(x + \mathtt{K})\,. \tag{3}$$

We claim that the variables y and z are "independent" elements of $\mathbb{F}_q$, in the sense that given x and y and assuming $\mathtt{K}$ is secret, it is *computationally infeasible* to (partially) find z or any information regarding it, and vice versa.

Claim. Let $\mathtt{K} \xleftarrow{\$} \mathbb{F}_q$ be a uniform random drawing secret key. Let $\mathcal{P}_{r-1}, \mathcal{P}_{r-2}, \ldots, \mathcal{P}_1, \mathcal{P}_0$ be $r \gg 1$ non-trivial permutations over $\mathbb{F}_q$ so that $x \mapsto \mathcal{P}_{\mathtt{K}}(x)$ defined as in Eq. (2) is indistinguishable from a PRP. Given an arbitrary $x \in \mathbb{F}_q$, and $y = P_{\mathtt{K}}(x) \in \mathbb{F}_q$ defined as before, the probability of any distinguisher $\mathcal{D}$ (with query and time complexity at most equal to q) to distinguish between $z \in \mathbb{F}_q$ as defined in Eq. (3) and a random $z' \xleftarrow{\$} \mathbb{F}_q$ is *negligible.*

We point out that for each fixed key $\mathtt{K}$, a collision can occur in z for two different inputs, but it can never occur in y. Further, computing z requires only a negligible amount of additional linear operations. Hence, we propose to make

$\mathcal{P}^{(e)}_{\mathtt{K}}$ dependent on both y and z, in order to achieve the goal of increasing the number of variables "for free".

For completeness, we mention that the idea of re-using the internal computation of an iterated function/permutation is not new in the literature. In [48], for example, a similar idea to set up a PRF is exploited. Given an iterated cipher E_k of r-rounds, let $E_k = E^{(2)}_k \circ E^{(1)}_k$, where $E^{(1)}_k$ denotes the first $r^{'}$ rounds and $E^{(2)}_k$ denotes the last $r^{''}$ rounds such that $r = r^{'} + r^{''}$ (usually $r' \approx r/2$). In there, Mennink and Neves proposed to set up a PRF of the form $x \mapsto E_k(x) + E^{(1)}_k(x)$, where E_k and $E^{(1)}_k$ are treated as two independent permutations. Later on, a similar approach has been exploited in the Fork design strategy [6].

3.3 The Megafono Design Strategy

Let $1 \le s' < s$. Let

– $\mathcal{P} : \mathbb{F}_{p^s} \to \mathbb{F}_{p^s}$ and $\mathcal{P}^{(e)}_{\mathtt{K}} : \mathbb{F}_{p^{2s}} \to \mathbb{F}_{p^{2s}}$ be two permutations;
– $\mathcal{R}_i : \mathbb{F}_{p^s} \to \mathbb{F}_{p^s}$ be the i-th rolling function, for each $i \ge 0$.

Let $\mathcal{F}^{(e)}_{\mathtt{K}} : (\mathbb{F}_{p^s})^2 \times \mathbb{N} \to (\mathbb{F}_{p^s})^3$ be defined as

$$\mathcal{F}^{(e)}_{\mathtt{K}}(y,z,i) \coloneqq \mathcal{T}^{L}_{2s,s+s'} \circ \mathcal{P}^{(e)}_{\mathtt{K}}(y, \mathcal{R}_i(z), i) \Big\| \mathcal{T}^{R}_{2s,s'} \circ \mathcal{P}^{(e)}_{\mathtt{K}}(y, \mathcal{R}_i(z), i) +$$

$$+ \mathcal{T}^{L}_{2s,s'} \circ \mathcal{P}^{(e)}_{\mathtt{K}}(y, \mathcal{R}_{i+1}(z), i+1) \Big\| \mathcal{T}^{R}_{2s,s+s'} \circ \mathcal{P}^{(e)}_{\mathtt{K}}(y, \mathcal{R}_{i+1}(z), i+1).$$

Based on all the considerations just given, $\textsc{Megafono}_{\mathtt{K}} : \mathbb{F}_{p^s} \to \mathbb{F}^{\star}_{p}$ is defined as

$$\textsc{Megafono}_{\mathtt{K}}(x) \coloneqq \mathcal{F}^{(e)}_{\mathtt{K}}(y,z,0) \mid\mid \mathcal{F}^{(e)}_{\mathtt{K}}(y,z,2) \mid\mid \cdots \mid\mid \mathcal{F}^{(e)}_{\mathtt{K}}(y,z,2\cdot i) \mid\mid \cdots ,$$

where $y, z \in \mathbb{F}_{p^s}$ are defined as in Eq. (3), i.e., $y = \mathcal{P}_{\mathtt{K}}(x) \coloneqq \mathtt{K} + \mathcal{P}(x + \mathtt{K})$ and z is the sums of all intermediate values of $\mathcal{P}_{\mathtt{K}}$. We emphasize that the rolling function is applied only on the variable z, so that a collision at the input of $\mathcal{P}^{(e)}_{\mathtt{K}}$ can never occur (remember that y is the output of a permutation).[9]

4 Specification of Hydra

4.1 The PRF Hydra

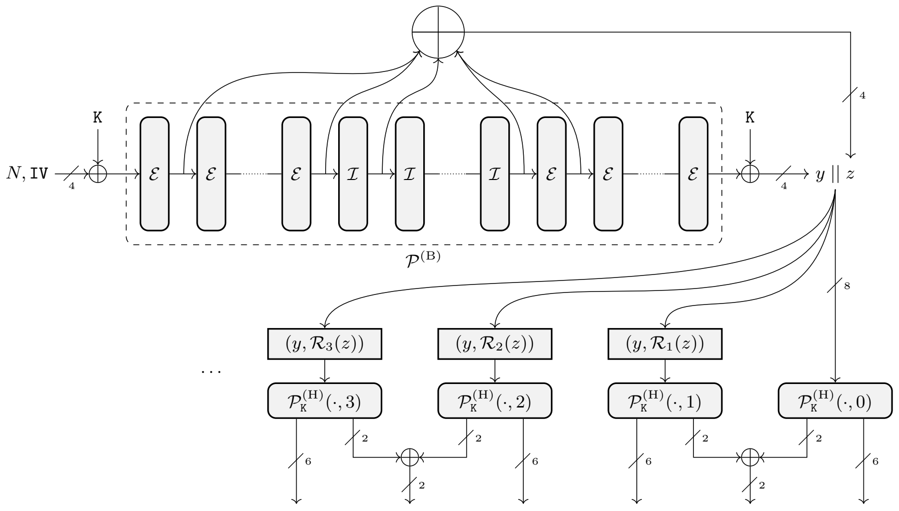
Let $p > 2^{63}$ (i.e., $\lceil \log_2(p) \rceil \ge 64$) and let $t \ge 4$ be the size of the output. Let κ be the security level such that $2^{80} \le 2^{\kappa} \le \min\{p^2, 2^{256}\}$. Let $\mathtt{K} \in \mathbb{F}^4_p$ be the master key. Let $2^{40} \le 2^{\kappa/2} \le \min\{p, 2^{128}\}$ be the data limit available for the attack.

Given a plaintext $P \in \mathbb{F}^t_p$, the ciphertext is defined by

$$C = \widehat{\mathcal{H}}([N \mid\mid \mathtt{IV}]) + P,$$

where $\widehat{\mathcal{H}} : \mathbb{F}^4_p \to \mathbb{F}^t_p$ is the $\textsc{Hydra}$ PRF, $\mathtt{IV} \in \mathbb{F}^3_p$ is a fixed initial value and $N \in \mathbb{F}_p$ is a nonce (e.g., a counter).

[9] We point out that a collision at the input of any $\mathcal{P}^{(e)}$ is possible in Farfalle/Ciminion, but the probability of such event is (much) smaller than the security level.



$N, \mathtt{IV}$
K
K
$\mathcal{E}$
$\mathcal{I}$
$y \,||\, z$
$\mathcal{P}^{(\mathrm{B})}$
$(y, \mathcal{R}_3(z))$
$(y, \mathcal{R}_2(z))$
$(y, \mathcal{R}_1(z))$
$\mathcal{P}^{(\mathrm{H})}_{\mathtt{K}}(\cdot, 3)$
$\mathcal{P}^{(\mathrm{H})}_{\mathtt{K}}(\cdot, 2)$
$\mathcal{P}^{(\mathrm{H})}_{\mathtt{K}}(\cdot, 1)$
$\mathcal{P}^{(\mathrm{H})}_{\mathtt{K}}(\cdot, 0)$
...
4
8
2
6

Fig. 3: The Hydra PRF.

Hydra. An overview of Hydra $\widehat{\mathcal{H}} : \mathbb{F}_p^4 \to \mathbb{F}_p^t$ is given in Fig. 3,[10] where

(1) $y = \mathtt{K} + \mathcal{P}^{(\mathrm{B})}([N \,||\, \mathtt{IV}] + \mathtt{K})$ for a certain permutation $\mathcal{P}^{(\mathrm{B})} : \mathbb{F}_p^4 \to \mathbb{F}_p^4$ defined in the following,

(2) $\mathcal{P}^{(\mathrm{H})}_{\mathtt{K}} : \mathbb{F}_p^8 \to \mathbb{F}_p^8$ is a keyed permutation defined in the following, and

(3) $\mathcal{R}_i(z) \equiv \mathcal{R}(z, i) : \mathbb{F}_p^4 \times \mathbb{N} \to \mathbb{F}_p^4$ is an affine function defined as $\mathcal{R}(z, i) = M_{\mathcal{R}}^i \times z$, where $M_{\mathcal{R}} \in \mathbb{F}_p^{4\times 4}$ is an invertible matrix that does not admit any (invariant) subspace trails, i.e., for each subspace $\mathfrak{U} \subset \mathbb{F}_p^4$ and for each $i \geq 0$ $\dim\left(\mathfrak{U} \cap (M_{\mathcal{R}}^i \times \mathfrak{U})\right) \leq \dim(\mathfrak{U}) - 1$.

We give an algorithmic description of Hydra in Algorithm 1 in Appendix C.1.

4.2 The Body of the Hydra: The Permutation $\mathcal{P}^{(\mathrm{B})}$

The permutation $\mathcal{P}^{(\mathrm{B})} : \mathbb{F}_p^4 \to \mathbb{F}_p^4$ is defined as

$$\mathcal{P}^{(\mathrm{B})}(\cdot) = \underbrace{\mathcal{E}_7 \circ \cdots \circ \mathcal{E}_4}_{4 \text{ times}} \circ \underbrace{\mathcal{I}_{R-1} \circ \cdots \circ \mathcal{I}_0}_{R \text{ times}} \circ \underbrace{\mathcal{E}_3 \circ \cdots \circ \mathcal{E}_0}_{4 \text{ times}}(M_{\mathcal{E}} \times \cdot), \tag{4}$$

where the external/internal rounds $\mathcal{E}_i, \mathcal{I}_j : \mathbb{F}_p^4 \to \mathbb{F}_p^4$ are defined as

$$\mathcal{E}_i(\cdot) = \varphi^{(\mathcal{E},i)} + M_{\mathcal{E}} \times S_{\mathcal{E}}(\cdot), \quad \mathcal{I}_j(\cdot) = \varphi^{(\mathcal{I},j)} + M_{\mathcal{I}} \times S_{\mathcal{I}}(\cdot)$$

for $i \in \{0, 1, \ldots, 7\}$ and each $j \in \{0, 1, \ldots, R-1\}$ where $\varphi^{(\mathcal{E},i)}, \varphi^{(\mathcal{I},j)} \in \mathbb{F}_p^4$ are randomly chosen round constants.

[10] The (Lernaean) Hydra is a mythological serpentine water monster with many heads. In our case, we can see $\mathcal{P}^{(\mathrm{B})}$ as the body of the Hydra, and the multiple parallel permutations $\mathcal{P}^{(\mathrm{H})}_{\mathtt{K}}$ as its multiple heads.

The Round Function $\mathcal{E}$. Let $d \geq 5$ be the smallest odd integer such that $\gcd(d, p^2 - 1) = 1$.[11] Let $\alpha, \alpha' \in \mathbb{F}_p \setminus \{0\}$ be arbitrarily chosen. The nonlinear layer $S_{\mathcal{E}} : \mathbb{F}_p^4 \to \mathbb{F}_p^4$ is defined as

$$S_{\mathcal{E}}(x_0, x_1, x_2, x_3) \coloneqq S_{\mathcal{E}}^{(1)}(x_0, x_1) \,||\, S_{\mathcal{E}}^{(2)}(x_2, x_3),$$

where $S_{\mathcal{E}}^{(1)}(x_0, x_1) = (y_0, y_1)$ and $S_{\mathcal{E}}^{(2)}(x_2, x_3) = (y_2, y_3)$ such that

$$y_i = \begin{cases} x_i \cdot \mathcal{D}'_{d,\alpha}(x_0 + x_1) & \text{if } i \in \{0, 1\}, \\ x_i \cdot \mathcal{D}'_{d,\alpha'}(x_2 - x_3) & \text{if } i \in \{2, 3\}, \end{cases}$$

and where

$$\mathcal{D}'_{d,\alpha}(x) \coloneqq \frac{\mathcal{D}_{d,\alpha}(x)}{x} \equiv \sum_{j=0}^{\lfloor d/2 \rfloor} \frac{d}{d-j} \cdot \binom{d-j}{j} \cdot (-\alpha)^j \cdot x^{d-2j-1} \tag{5}$$

is defined via the Dickson polynomial $\mathcal{D}_{d,\alpha}(\cdot)$ (see Definition 2). As we are going to show, the function $S_{\mathcal{E}}$ is invertible.

The invertible matrix $M_{\mathcal{E}} \in \mathbb{F}_p^{4\times 4}$ is an MDS matrix such that no (invariant) subspace trails generated by $\langle (1, -1, 0, 0) \rangle \subseteq \mathbb{F}_p^4$ and $\langle (0, 0, 1, 1) \rangle \subseteq \mathbb{F}_p^4$ exist (see Appendix D.2 for more details). For our purposes, we choose the AES matrix, i.e., $M_{\mathcal{E}} = \mathrm{circ}(3, 2, 1, 1)$.

The Round Function $\mathcal{I}$. The nonlinear layer $S_{\mathcal{I}} : \mathbb{F}_p^4 \to \mathbb{F}_p^4$ is defined as

$$S_{\mathcal{I}}(x_0, x_1, x_2, x_3) = (x_0 + \hat{y}, x_1 + \hat{y}, x_2 + \hat{y}, x_3 + \hat{y}),$$

where $\hat{y} \in \mathbb{F}_p$ is defined as

$$\hat{y} = \left(\left(\sum_{j=0}^{3} \lambda_j^{(0)} x_j \right)^2 + \left(\sum_{j=0}^{3} \lambda_j^{(1)} x_j \right) + \lambda' \right) \cdot \left(\left(\sum_{j=0}^{3} \lambda_j^{(0)} x_j \right)^2 + \left(\sum_{j=0}^{3} \lambda_j^{(1)} x_j \right) + \lambda'' \right)$$

such that

(1) $\lambda_0^{(0)}, \dots, \lambda_3^{(0)}, \lambda_0^{(1)}, \dots, \lambda_3^{(1)}, \lambda', \lambda'' \in \mathbb{F}_p \setminus \{0\}$,
(2) $\sum_{j=0}^{3} \lambda_j^{(0)} = \sum_{j=0}^{3} \lambda_j^{(1)} = 0$, and
(3) $\{\lambda_j^{(0)}\}_{j=0}^{3}$ and $\{\lambda_j^{(1)}\}_{j=0}^{3}$ are linearly independent.

The invertible matrix $M_{\mathcal{I}} \in \mathbb{F}_p^{4\times 4}$ must provide full diffusion and must satisfy the following conditions:

(i) for $i \in \{0, 1\}$: $\quad \sum_{j=0}^{3} \lambda_j^{(i)} \cdot \left(\sum_{l=0}^{3} M_{\mathcal{I}}[j, l] \right) \neq 0$,
(ii) for $i \in \{0, 1\}$ and for $j \in \{0, 1, \dots, 3\}$: $\quad \sum_{l=0}^{3} \lambda_l^{(i)} \cdot M_{\mathcal{I}}[l, j] \neq 0$, and

[11] Note that $d = 3$ does not satisfy $\gcd(d, p^2 - 1) = 1$ for any $p \geq 3$.

(iii) it prevents infinitely long subspace trails for the generalized Lai–Massey construction $S_{\mathcal{I}}$ (details are given in Appendix F).

In particular, we suggest using a matrix of the form

$$M_{\mathcal{I}} = \begin{pmatrix} \mu_{0,0}^{(\mathcal{I})} & 1 & 1 & 1 \\ \mu_{1,0}^{(\mathcal{I})} & \mu_{1,1}^{(\mathcal{I})} & 1 & 1 \\ \mu_{2,0}^{(\mathcal{I})} & 1 & \mu_{2,2}^{(\mathcal{I})} & 1 \\ \mu_{3,0}^{(\mathcal{I})} & 1 & 1 & \mu_{3,3}^{(\mathcal{I})} \end{pmatrix} \tag{6}$$

where the values $\mu_{i,j}^{(\mathcal{I})} \in \mathbb{F}_p \setminus \{0\}$ are found using the tool given in Appendix F.2, in order to satisfy conditions *(i)* – *(iii)* above.

4.3 The Heads of the Hydra: The Permutation $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$

The keyed permutation $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})} : (\mathbb{F}_p^4)^2 \times \mathbb{N} \to \mathbb{F}_p^8$ is defined as

$$\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, i) = \underbrace{\mathtt{K}' + \mathcal{J}_{R_{\mathrm{H}}-1} \circ (\mathtt{K}' + \mathcal{J}_{R_{\mathrm{H}}-2}) \circ \ldots \circ (\mathtt{K}' + \mathcal{J}_1) \circ (\mathtt{K}' + \mathcal{J}_0)}_{R_{\mathrm{H}} \text{ times}}(\cdot, i)$$

where $\mathtt{K}' = \mathtt{K} \,||\, (\mathrm{circ}(3,2,1,1) \times \mathtt{K}) \in \mathbb{F}_p^8$. For $j \in \{0, 1, \ldots, R_{\mathrm{H}} - 1\}$, we define $\mathcal{J}_j(\cdot, i) : \mathbb{F}_p^8 \to \mathbb{F}_p^8$ as

$$\mathcal{J}_j(\cdot, i) = \varphi_{j,i} + M_{\mathcal{J}}^{(i \mod 2)} \times S_{\mathcal{J}}^{(i,j)}(\cdot),$$

where $\varphi_{j,i} \in \mathbb{F}_p^8$ are random round constants.

The nonlinear layer $S_{\mathcal{J}}^{(i,j)}(x_0, x_1, \ldots, x_7) = (y_0, \ldots, y_7)$ is defined by

$$y_l = x_l + \psi_{i,j} \cdot \left(\psi'_{i,j} + \sum_{h=0}^{7} \lambda_h^{(i \mod 2)} \cdot x_h \right)^2 \quad \text{for } 0 \le l \le 7,$$

where

(1) $\psi'_{i,j}, \psi_{i,j} \in \mathbb{F}_p \setminus \{0\}$ are random round constants, and
(2) for $i \in \{0,1\}$, $\lambda_0^{(i)}, \ldots, \lambda_7^{(i)} \in \mathbb{F}_p \setminus \{0\}$ are arbitrary s.t. $\sum_{h=0}^{7} \lambda_h^{(i)} = 0$.

We also assume that $\lambda_0^{(0)}, \ldots, \lambda_7^{(0)}$ and $\lambda_0^{(1)}, \ldots, \lambda_7^{(1)}$ are linearly independent. The invertible matrices $M_{\mathcal{J}}^{(i)} \in \mathbb{F}_p^{8\times 8}$ must fulfill similar conditions to *(i)* – *(iii)* described in Section 4.2 for each $i \in \{0,1\}$. That is, for each $i \in \{0,1\}$: *(i)* $\sum_{h=0}^{7} \lambda_h^{(i)} \left(\sum_{l=0}^{7} M_{\mathcal{J}}^{(i)}[h,l] \right) \neq 0$; *(ii)* $\sum_{l=0}^{7} \lambda_l^{(i)} \cdot M_{\mathcal{J}}^{(i)}[l,h] \neq 0$, for $h \in 0, \ldots, 7$; and *(iii)* $M_{\mathcal{J}}^{(i)}$ prevents infinitely long subspace trails related to $\lambda_0^{(i)}, \ldots, \lambda_7^{(i)}$, [12] as detailed in Appendix F. We recommend that $M_{\mathcal{J}}^{(i)}$ has a similar form to the matrix in Eq. (6) extended to eight rows and columns.

[12] Note that if the scheme instantiated with $\{\lambda_h\}_{h=0}^{7}$ is secure against invariant subspaces, then it is also secure when instantiated with $\{\psi \cdot \lambda_h\}_{h=0}^{7}$ for each $\psi \in \mathbb{F}_p \setminus \{0\}$.

4.4 Number of Rounds

In order to provide κ bits of security (with a data limit of $2^{\kappa/2}$), the number of rounds must satisfy

$$\begin{aligned} R_{\mathcal{I}} &= \left\lceil 1.125 \cdot \left\lceil \max\left\{ \frac{\kappa}{4} - \log_2(d) + 3, \widehat{R_{\mathcal{I}}} \right\} \right\rceil \right\rceil , \\ R_{\text{H}} &= \left\lceil 1.25 \cdot \left\lceil \max\left\{ 24, \widehat{R_{\text{H}}}, 3 + R_{\text{H}}^{*} \right\} \right\rceil \right\rceil , \end{aligned}$$

where $\widehat{R_{\mathcal{I}}}$, $\widehat{R_{\text{H}}}$, and R_{H}^{*} are the minimum positive integers that satisfy Eq. (14), Eq. (16), and Eq. (10) respectively, and where we added a security margin of 12.5% and 25% respectively. In Appendix A, we provide a script that given p and κ, returns the number of rounds $R_{\mathcal{I}}$ and R_{H}. E.g., $R_{\mathcal{I}} = R_{\text{H}} = 38$ for $p \approx 2^{128}$ and $\kappa = 128$.

Finally, we refer to Appendix C.2 for details on how we generate the pseudo-random constants. We also emphasize that we do *not* claim security against related-key attacks.

5 From Lai–Massey to the `Amaryllises` Scheme

5.1 The Lai–Massey Scheme

The Lai–Massey scheme [43] over $\mathbb{F}_q^2$ is defined as $(x, y) \mapsto (x + F'(x - y), y + F'(x - y))$, where $F' : \mathbb{F}_q \to \mathbb{F}_q$. Two generalizations over $\mathbb{F}_p^n$ have recently been proposed in [29], defined as $(x_0, x_1, \ldots, x_{n-1}) \mapsto (y_0, y_1, \ldots, y_{n-1})$, where

(1) $y_i = x_i + \sum_{j=0}^{n-1} F'(x_j - x_{j-1})$ for $i \in \{0, 1, \ldots, n-1\}$, and any $n \geq 3$; or

(2) $y_i = x_i + F'\left(\sum_{j=0}^{n-1} (-1)^j \cdot x_j\right)$ for $i \in \{0, 1, \ldots, n-1\}$, and $n \geq 3$ even.

These schemes can be further generalized as described in the following.

Proposition 1. *Let $q = p^s$, where $p \geq 3$ is a prime and s is a positive integer, and let $n \geq 2$. Given $1 \leq l \leq n-1$, let $\lambda_0^{(i)}, \lambda_1^{(i)}, \ldots, \lambda_{n-1}^{(i)} \in \mathbb{F}_q$ be such that $\sum_{j=0}^{n-1} \lambda_j^{(i)} = 0$ for $i \in \{1, 2, \ldots, l\}$. Let $F' : \mathbb{F}_q^l \to \mathbb{F}_q$. The function $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ defined as $\mathcal{F}(x_0, \ldots, x_{n-1}) = (y_0, \ldots, y_{n-1})$ is invertible when*

$$y_l = x_l + F'\left(\sum_{j=0}^{n-1} \lambda_j^{(1)} \cdot x_j, \sum_{j=0}^{n-1} \lambda_j^{(2)} \cdot x_j, \ldots, \sum_{j=0}^{n-1} \lambda_j^{(l)} \cdot x_j \right), \textit{ for } 0 \leq l \leq n-1.$$

No conditions are imposed on F'. Even if not strictly necessary, it make sense to choose $\{\lambda_j^{(1)}\}_{j=0}^{n-1}, \{\lambda_j^{(2)}\}_{j=0}^{n-1}, \ldots, \{\lambda_j^{(l)}\}_{j=0}^{n-1}$ such that they are linearly independent. Since $\sum_{j=0}^{n-1} \lambda_j^{(i)} = 0$ for $i \in \{1, 2, \ldots, l\}$, there are at most $n-1$ linearly independent combinations. For a proof we refer to Appendix G.1.

5.2 The `Amaryllises` Scheme

Next, we introduce the `Amaryllises` scheme, in which the sum operation in the Lai–Massey scheme is replaced by a multiplication.

Theorem 1 (The `Amaryllises` Scheme.). *Let $q = p^s$, where $p \geq 3$ is a prime and s is a positive integer, and let $t \geq 2$ be an integer. Let*

1. *$F : \mathbb{F}_q \to \mathbb{F}_q$ be a function such that (1st) $F(0) \neq 0$ and (2nd) $G(x) \coloneqq x \cdot F(x)$ is invertible over $\mathbb{F}_q$;*
2. *$H : \mathbb{F}_q \to \mathbb{F}_q$ be any function;*
3. *$\beta_0, \beta_1, \ldots, \beta_{n-1} \in \mathbb{F}_q \setminus \{0\}$ such that $\sum_{i=0}^{n-1} \beta_i = 0$ **if** there exists (at least) one $x \in \mathbb{F}_p$ such that $H(x) \neq 0$ (equivalently, no condition on $\sum_{i=0}^{n-1} \beta_i$ is imposed if $H(x) = 0$ for each $x \in \mathbb{F}_q$).*

The `Amaryllises` construction $x \mapsto y$ over $\mathbb{F}_q^t$ defined as

$$y_i = x_i \times F\left(\sum_{i=0}^{n-1} \beta_i \cdot x_i\right) + H\left(\sum_{i=0}^{n-1} \beta_i \cdot x_i\right)$$

for $i \in \{0, 1, \ldots, n-1\}$ is invertible.

Proof. First of all, we prove that $F(z) \neq 0$ for each $z \in \mathbb{F}_q$. Since G is a permutation and since $G(0) = F(0) \cdot 0 = 0$ by definition, then $G(x) \neq 0$ for each $x \neq 0$. It follows that $F(x) = G(x)/x \neq 0$ for any $x \in \mathbb{F} \setminus \{0\}$, while $F(0) \neq 0$ by assumption.

The invertibility of `Amaryllises` follows from the fact that

$$\begin{aligned}\sum_{i=0}^{n-1} \beta_i \cdot y_i = &\left(\sum_{i=0}^{n-1} \beta_i \cdot x_i\right) \times F\left(\sum_{i=0}^{n-1} \beta_i \cdot x_i\right) + H\left(\sum_{i=0}^{n-1} \beta_i \cdot x_i\right) \times \underbrace{\sum_{i=0}^{n-1} \beta_i}_{=0} \\ =&G\left(\sum_{i=0}^{n-1} \beta_i \cdot x_i\right),\end{aligned}$$

where G is invertible by definition. Note that the condition $\sum_{i=0}^{n-1} \beta_i = 0$ is not necessary if $H(x) = 0$ for each $x \in \mathbb{F}_q$. It follows that

$$\sum_{i=0}^{n-1} \beta_i \cdot x_i = G^{-1}\left(\sum_{i=0}^{n-1} \beta_i \cdot y_i\right) \qquad \Longrightarrow \qquad x_i = \frac{y_i - H\left(G^{-1}\left(\sum_{i=0}^{n-1} \beta_i \cdot y_i\right)\right)}{F\left(G^{-1}\left(\sum_{i=0}^{n-1} \beta_i \cdot y_i\right)\right)},$$

where $F(z) \neq 0$ for each z. □

5.3 Constructing F as in Theorem 1

Here we show how to construct functions F that satisfy the assumptions of Theorem 1 and that can be efficiently computed.

Constructing F via Power Maps. Let $d \geq 3$ be the smallest integer such that $x \mapsto x^d$ is invertible over $\mathbb{F}_q$, hence $\gcd(d, q-1) = 1$. By applying the change of variable $x = z - \alpha$ for $\alpha \in \mathbb{F}_q \setminus \{0\}$, we get $x^d = (z \pm \alpha)^d = \sum_{i=0}^{d} \binom{d}{i} \cdot z^i \cdot (\pm\alpha)^{d-i}$. Let F over $\mathbb{F}_q$ be defined as

$$F(z) = \frac{(z \pm \alpha)^d \mp \alpha^d}{z} = \sum_{i=1}^{d} \binom{d}{i} z^{i-1} \cdot (\pm\alpha)^{d-i}\,. \qquad (7)$$

F satisfies the requirements of Theorem 1, i.e., $F(0) = \pm d \cdot \alpha^{d-1} \neq 0$ (since $\alpha \neq 0$) and $F(z) \cdot z = (z \pm \alpha)^d \mp \alpha^d$ is invertible since $x \mapsto x^d$ is invertible.

Prime Case – Dickson Polynomials. Dickson polynomials generalize the power maps $x \mapsto x^d$, that is, $\mathcal{D}_{d,0}(x) = x^d$.

Definition 2 (Dickson Polynomials). *Let $q = p^s$, where $p \geq 3$ is a prime and s is a positive integer. Let $\alpha \in \mathbb{F}_q$ fixed. The Dickson polynomial $\mathcal{D}_{d,\alpha}(x)$ of degree d over $\mathbb{F}_q$ is defined as*

$$\mathcal{D}_{d,\alpha}(x) \coloneqq \sum_{j=0}^{\lfloor d/2 \rfloor} \frac{d}{d-j} \cdot \binom{d-j}{j} \cdot (-\alpha)^j \cdot x^{d-2j}.$$

For the following, note that if d is odd, only monomials with odd exponents appear, and if d is even, only monomials with even exponents appear.

Theorem 2 ([50]). *Let $q = p^s$ as before. Let $\alpha \in \mathbb{F}_q$. The Dickson polynomial $\mathcal{D}_{d,\alpha}(x)$ is a permutation polynomial of $\mathbb{F}_q$ if and only if $\gcd(d, q^2-1) = 1$. The inverse of $\mathcal{D}_{d,\alpha}$ is the Dickson polynomial $\mathcal{D}_{\hat{d},\alpha^d}$ where $d \cdot \hat{d} = 1 \mod q^2 - 1$.*

Based on this, we can prove the following (see Appendix G.2 for details).

Lemma 1. *Let $q = p^s$, where $p \geq 3$ is a prime and s is a positive integer. Let $d \geq 5$ be the smallest odd integer such that $\gcd(d, q^2-1) = 1$. Let $\alpha, \beta_0, \ldots, \beta_{t-1} \in \mathbb{F}_q \setminus \{0\}$. The nonlinear layer $\mathcal{J} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ defined as $\mathcal{J}(x_0, x_1, \ldots, x_{n-1}) = (y_0, y_1, \ldots, y_{n-1})$ where $y_i = x_i \cdot \mathcal{D}'_{d,\alpha}(\beta_0 \cdot x_0 + \beta_1 \cdot x_1 + \ldots + \beta_{n-1} \cdot x_{n-1})$ and where $\mathcal{D}'_{d,\alpha}$ is defined as in Eq. (5) is invertible.*

To prove such result, it is sufficient to show that $\mathcal{D}'_{d,\alpha}$ satisfies the requirements of Theorem 1. Note that $\mathcal{D}'_{d,\alpha}(x)$ can be computed via $\frac{d-1}{2}$ $\mathbb{F}_q$-multiplications, with respect to $d-1$ $\mathbb{F}_q$-multiplications required to compute the function given in Eq. (7).

6 Design Rationale of $\mathcal{P}^{(\mathrm{B})}$, $\mathcal{R}_i$ and of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$

Initial Considerations. We start by pointing out the following.

– The Even–Mansour construction $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ guarantees $2 \cdot \log_2(p) \geq \kappa$ bits of security [19, 25] since $\mathcal{P}^{(\mathrm{B})}$ is designed in order to resemble a PRP.

– Since $2^\kappa \leq p^2$, we have to truncate at least two $\mathbb{F}_p$ elements in order to prevent trivial brute-force attacks (i.e., guessing of the truncated part), and therefore we chose this amount.

The keyed permutation $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ is defined over $\mathbb{F}_p^8$ in order to take into account $y \in \mathbb{F}_p^4$ and $z \in \mathbb{F}_p^4$ defined as before.

6.1 The Hades Design Strategy

In order to design $\mathcal{P}^{(\mathrm{B})}$, we aim to retain the advantages of the Hades strategy [28], in particular the security arguments against statistical attacks and the efficiency of the partial middle rounds. The Hades strategy is a way to design SPN schemes over $\mathbb{F}^t$ in which rounds with *full S-box layers* are mixed with rounds with *partial S-box layers*. The external rounds with full S-box layers (that is, t S-boxes in each nonlinear layer) at the beginning and at the end of the construction ensure security against statistical attacks. The rounds with partial S-box layers (that is, $t' < t$ S-boxes and $t - t'$ identity functions) in the middle of the construction are cheaper to evaluate in e.g. MPC settings, and help to prevent algebraic attacks. In all rounds, the linear layer is defined via the multiplication of a MDS matrix.

This strategy has been recently pushed to its limit in Neptune [29]. In such a case, instead of using the same matrix and the same S-box both for the external and the internal rounds, Neptune's designers propose to use two different S-boxes and two different matrices, one dedicate for the external rounds and one dedicate for the internal ones.

6.2 The External Rounds of $\mathcal{P}^{(\mathrm{B})}$: The Nonlinear Layer $S_{\mathcal{E}}$

As in Hades, Poseidon, and Neptune, we use the external rounds to provide security against statistical attacks. In the case of Hades and Poseidon, this is achieved by instantiating the external full rounds with power maps $x \mapsto x^d$ for each of the t words. Since computing $x \mapsto x^d$ requires $\mathrm{hw}(d) + \lfloor \log_2(d) \rfloor - 1 \geq 2$ multiplications, the total number of multiplications for this nonlinear layer is $t \cdot (\mathrm{hw}(d) + \lfloor \log_2(d) \rfloor)$, i.e., at least 2 multiplications per word.[13] Moreover, no diffusion among the words takes place.

The `Amaryllises` scheme from Theorem 1 allows to solve these issues. As we have already seen, the best choice for minimizing the number of multiplications is to instantiate F in Theorem 1 with a Dickson polynomial (precisely, with $\mathcal{D}'_{d,\alpha}$) and to fix $H = 0$. The only drawback is the existence of invariant subspaces [45, 46] for this function (these would exist even if $H \neq 0$). E.g., by instantiating $S_{\mathcal{E}}$ with $y_i = x_i \cdot \mathcal{D}'_{d,\alpha}(\beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)$ for $i \in \{0, 1, 2, 3\}$, the round

[13] Given $d = \sum_{i=0}^{\lfloor \log_2(d) \rfloor} d_i \cdot 2^i$ for $d_i \in \{0, 1\}$, computing $x \mapsto x^d$ requires computing x^{2^j} for each $j \in \{0, 1, \ldots, \lfloor \log_2(d) \rfloor\}$ for a cost of $\lfloor \log_2(d) \rfloor$ multiplications, plus $\mathrm{hw}(d) - 1$ multiplications to get $x \mapsto x^d$ ($\mathrm{hw}(\cdot)$ is the Hamming weight).

function $\mathcal{E}$ admits

$$\mathfrak{D}_r = \left\{ (x_0, \ldots, x_3) \in \mathbb{F}_p^4 \,\middle|\, \forall l \in \{0, \ldots, r-1\} : \sum_{i=0}^{3} \beta_i \cdot \left(\sum_{j=0}^{3} (M_{\mathcal{E}}[i,j])^l \cdot x_j \right) = 0 \right\}$$

as an invariant subspace for $r \geq 1$ rounds (where $(M_{\mathcal{E}})^0$ is the identity matrix). Independently of the linear layer, this subspace covers (at least) three rounds. To solve this issue, besides choosing a suitable matrix, we decided to set up $S_{\mathcal{E}}$ via the concatenation of two independent S-boxes, each one defined via the `Amaryllises` scheme. Compared to a nonlinear layer built with independent power maps, diffusion also takes place among the different words and a higher degree can be reached for the same multiplicative cost.

6.3 The Internal Rounds of $\mathcal{P}^{(\mathbf{B})}$

The main goal of the internal rounds is to increase the overall degree of the function. In HadesMiMC and Poseidon, the nonlinear layer in the middle rounds are instantiated with a single power map $x \mapsto x^d$ and $(t-1)$ identity functions. As mentioned in the introduction, the cost metric in an MPC application is related to the number of nonlinear operations, i.e., multiplications over $\mathbb{F}_p$.

To minimize this number, it could be desirable to use only one multiplication per round. Indeed, let us compare the cost in terms of $\mathbb{F}_p$ multiplications in order to reach a certain degree Δ when using a round instantiated with the quadratic map $x \mapsto x^2$ versus one instantiated via an invertible power map $x \mapsto x^d$ with $d \geq 3$. By comparing the overall number of $\mathbb{F}_p$ multiplications, the first option is always the most competitive, since

$$\underbrace{\lceil \log_2(\Delta) \rceil = \lceil \log_d(\Delta) \cdot \log_d(2) \rceil}_{\text{using } x \mapsto x^2} < \underbrace{\lceil \log_d(\Delta) \rceil \cdot (\lfloor \log_2(d) \rfloor + \text{hw}(d) - 1)}_{\text{using } x \mapsto x^d},$$

where $\lceil \log_d(\Delta) \cdot \log_d(2) \rceil < \lceil \log_d(\Delta) \rceil \cdot \lceil \log_d(2) \rceil$ and where $\lfloor \log_2(d) \rfloor + \text{hw}(d) - 1 \geq \lfloor \log_2(d) \rfloor + 1 = \lceil \log_2(d) \rceil$. As a concrete example, let us focus on the case $d = 3$ and $\Delta = 2^{128}$. In the first case, 128 rounds and 128 $\mathbb{F}_p$ multiplications are required to reach the degree Δ. In the second case, 81 rounds and 162 $\mathbb{F}_p$ multiplications are required, i.e., 21% fewer multiplications in total.

However, the function $x \mapsto x^2$ is not invertible, and a partial nonlinear layer instantiated with it could result in an insecure PRF. The problem can be solved by using the quadratic map in a mode that preserves the invertibility, as in a Feistel construction or in a generalized Lai–Massey construction as proposed in Section 5.1. In order to reduce the depth, we decided for a generalized Lai–Massey construction instantiated with nonlinear function of degree 4 that can be computed with 2 multiplications only.

As already pointed out e.g. in [52], invariant subspaces exist for the Lai–Massey construction. Hence, it is crucial to choose the matrix $M_{\mathcal{I}}$ in order to break them. In Appendix F, we show how to adapt the analysis/tool proposed in [32, 34] for breaking arbitrarily long subspace trails for P-SPN schemes to the case of the generalized Lai–Massey constructions.

Finally, we point out that the conditions *(i)* – *(ii)* in Section 4.2 are crucial to guarantee the density of the interpolation polynomial, as shown in Appendix E.1.

6.4 About $\mathcal{R}_i$ and $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$

As in the case of Farfalle and Ciminion, the attacker knows the outputs of the expansion phase of Megafono, but they cannot choose them (in order to set up e.g. a chosen-ciphertext attack). Further, they do not know (and so choose) its inputs. By designing $\mathcal{P}^{(\mathrm{B})}$ in order to resemble a PRP, the attacker is not able to choose texts at the input of $\mathcal{P}^{(\mathrm{B})}$ which result in texts with some specific statistical/algebraic properties at the inputs of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ (respectively, output of $\mathcal{P}^{(\mathrm{B})}$). Based on these considerations, it turns out that only few attacks could work at the expansion phase of Megafono, and so of Hydra. Since the possible attacks are mainly algebraic ones and since the overall cost of Hydra growths linearly with the cost of computing $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$, we decided to work with a generalized Lai–Massey construction of degree 2. This allows us to defeat possible attacks, and to get good overall performance, as we are going to show.

We decided to keep the rolling function $\mathcal{R}_i$ linear, contrary to what is done in e.g. Xoofff [20] and Ciminion. As already noted, due to the particular structure of Farfalle and of Megafono, an attacker can exploit the relation between consecutive evaluations of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ on the same output of $\mathcal{P}$ in order to break the scheme [15, 18]. Instead of dealing with (low-degree) nonlinear rolling functions, we frustrate such an attack strategy by (1st) working with the keyed permutation $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$, (2nd) truncating the final output, and (3rd) imposing that no (invariant) subspace trail exists, via the condition $\dim\big(\mathfrak{U} \cap (M_{\mathcal{R}}^{i} \times \mathfrak{U})\big) \leq \dim(\mathfrak{U}) - 1$ for each $\mathfrak{U} \subset \mathbb{F}_p^4$ and for each $i \geq 0$.

7 Security Analysis of $\mathcal{P}^{(\mathrm{B})}$

Attacks taking into account the relations between the inputs and the outputs of Hydra are in general harder than the attacks taking into account the relations between the inputs and the outputs of $\mathcal{P}^{(\mathrm{B})}$. In this section, we assume the attacker knows both the inputs and outputs of $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$. Note that this is an unrealistic but favorable scenario for the attacker, since the outputs of $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ are actually hidden from $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$. It follows that, if an attacker is not able to break $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ in a scenario in which they have full control over the inputs and outputs, they cannot break Hydra by exploiting the relation of its inputs and outputs.

The chosen number of rounds guarantees that $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ resembles a PRP against attacks with a computational complexity of at most 2^{κ} and with a data complexity of at most $2^{\kappa/2}$. We assume that the last two external rounds $\mathcal{E}_7$, $\mathcal{E}_6$ are part of the security margin (hence, not considered in our security analysis). Due to the page limit and the similarity between $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ and HadesMiMC/Poseidon, here we only highlight the main points of the security analysis of $\mathcal{P}^{(\mathrm{B})}$. We refer to Appendices D and E for all details.

Statistical Analysis and the External Rounds. Similar to HadesMiMC and Poseidon, we make use of the external rounds in order to provide security against statistical attacks. In particular, we claim the following.

Claim. Consider a permutation $\widehat{\mathcal{P}^{(\mathrm{B})}}$ over $\mathbb{F}_p^4$ defined as $\mathcal{P}^{(\mathrm{B})}$ in Eq. (4), albeit with the middle rounds replaced by an invertible linear layer $L_{\mathcal{P}^{(\mathrm{B})}} \in \mathbb{F}_p^{4\times 4}$ and where two external rounds are removed, i.e.,

$$\widehat{\mathcal{P}^{(\mathrm{B})}}(\cdot) := \underbrace{\mathcal{F} \circ \mathcal{F}}_{2 \text{ times}} \circ L_{\mathcal{P}^{(\mathrm{B})}} \circ \underbrace{\mathcal{F} \circ \cdots \circ \mathcal{F}}_{4 \text{ times}}(\cdot)\,.$$

The security against statistical attacks of $\mathcal{P}^{(\mathrm{B})}$ is not lower than that of $\widehat{\mathcal{P}^{(\mathrm{B})}}$.

In a differential attack [12, 13], given pairs of inputs with fixed input differences, differential attacks consider the probability distribution of the corresponding output differences produced by the cryptographic primitive. As our design is an iterated scheme, a cryptanalyst searches for ordered sequences of differences over any number of rounds that are called differential characteristics/trails. In Appendix D.1, we show that

– the maximum differential probability of the function $(x, y) \mapsto S_{\mathcal{E}}^{(i)}(x, y)$ for $i \in \{0, 1\}$ is d/p, and
– the probability of any differential characteristic over three pairs of consecutive rounds it is at most $(d/p)^9 \ll 2^{-2.5\cdot\kappa}$, where $p^{-2} \leq 2^{-\kappa}$ and $5 \leq d \ll p$.

Variants of differential attacks include truncated differentials [42], in which the attacker can specify only part of the difference between pairs of texts. In Appendix D.2, we show that the matrix $M_{\mathcal{E}}$ is chosen in order to destroy truncated differential with probability 1 (which correspond to the invariant subspaces $\langle(1, -1, 0, 0)\rangle \subseteq \mathbb{F}_p^4$ and $\langle(0, 0, 1, 1)\rangle \subseteq \mathbb{F}_p^4$ of the nonlinear layer).

Finally, other statistical attacks do not pose any threat to $\mathcal{P}^{(\mathrm{B})}$ due to the facts that (1st) full diffusion is achieved over a single round and (2nd) the arrangement/alignment of $S_{\mathcal{E}}$ and the one of $M_{\mathcal{E}}$ are different ($S_{\mathcal{E}}$ mixes two $\mathbb{F}_p$-elements in a nonlinear way, while $M_{\mathcal{E}}$ is defined as a matrix in $\mathbb{F}_p^{4\times 4}$ which does *not* admit an equivalent representation in $\mathbb{F}_{p^2}^{2\times 2}$).

Algebraic Analysis and the Internal Rounds. Regarding the internal rounds, we point out that no subspace trail [45, 46, 30] can cover an arbitrary number of rounds $\mathcal{I}$, due to the choice of the matrix $M_{\mathcal{I}}$, as explained in Appendix F. In there, we show how to adapt the analysis/tool proposed in [32, 34] for breaking arbitrarily long subspace trail for P-SPN schemes to the case of the generalized Lai–Massey constructions.

Algebraic attacks exploit the low degree and/or the sparsity of the system of equations that describes the scheme in order to break it. One of the most powerful algebraic attack is the interpolation attack [38], whose goal is to construct an interpolation polynomial that describes the function. The cost of setting up such an attack depends on the number of different monomials in the interpolation polynomial, where (an upper/lower bound of) the number of different monomials can be estimated via the degree of the function. For this purpose, in Appendix E.1, we show that the equations that describe $\mathcal{P}^{(\mathrm{B})}$ are dense and in Appendix E.2 we show that $R_{\mathcal{I}} \geq \frac{\kappa}{4} - \log_2(d) + 3$ rounds are necessary for preventing Meet-in-the-middle (MitM) interpolation attacks. In particular, only a few $\mathcal{E}$ rounds are sufficient to reach the maximum degree in the backward direction, since the inverse of the Dickson polynomial has a high degree (i.e., of the same order as p) if $d \ll p$ is a small integer (see Theorem 2).

Next, we show that the number of rounds necessary for preventing interpolation attacks are also sufficient to prevent Gröbner basis attack in Appendix E.4 (see also Section 8.2 for more information on Gröbner basis attacks). In a similar way, other algebraic attacks as the higher-order differential one [42, 11] do not pose any threat on the security of $\mathcal{P}^{(\mathrm{B})}$.

8 Security Analysis of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$

While $x \mapsto \mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ is designed to behave like a PRP (in a scenario in which an attacker can can choose its inputs/outputs), we do not impose such a strong condition for $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$. In order to be competitive in MPC scenarios, our strategy can be summarized as follows: *$\mathcal{P}_{K}^{(\mathrm{H})}$ is designed so that* Hydra *is secure under the assumption that $\mathtt{K} + \mathcal{P}^{(\mathrm{B})}(x + \mathtt{K})$ behaves like a PRP*. Hence, $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ will be studied assuming that the attacker only knows the outputs of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ after summation-truncation and that the attacker does not know the inputs of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$.

The attacker is still able to utilize that the outputs from a (potentially large) number of permutations $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ hail from the same unknown input $y, z \in \mathbb{F}_p^4$. This can, for instance, be used when constructing systems of polynomial equations from $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$. Indeed, we will later see that the most competitive attacks will be based on Gröbner bases.

8.1 Statistical and Invariant Subspace Attacks

Statistical attacks are particularly frustrated by the fact that it is infeasible for the attacker to choose a set of inputs $\{x_j\}_j$ for $\mathcal{P}^{(\mathrm{B})}$, such that the corresponding outputs $\{y_j\}_j$ satisfy certain statistical/algebraic properties that can be exploited to break $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$. Besides, we point out that the round constants $\varphi^{(j,i)} \in \mathbb{F}_p^8$ and the $\psi_{i,j} \in \mathbb{F}_p \setminus \{0\}$ change at every round j and at every consecutive $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ call. This design strategy was introduced in Rasta [23] (where the affine layer changes at every encryption) for preventing statistical attacks. In addition to these precautions, it is still desirable that $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ has good statistical properties.

To this end, the invertible matrices $M_{\mathcal{J}}^{(i \mod 2)} \in \mathbb{F}_p^{8\times 8}$ are chosen so that no invariant subspace trail can cover more than 7 rounds. We refer to Appendix F for more details. Based on this, the probability of each differential characteristic over R_{H} rounds is at most $p^{-\lfloor R_{\mathrm{H}}/8 \rfloor}$, since the maximum differential probability of $S_{\mathcal{J}}^{(i,j)}$ is p^{-1} (see Appendix H.1) and at least one $S_{\mathcal{J}}^{(i,j)}$ function is active every 8 rounds. By choosing $R_{\mathrm{H}} \geq 24$, the probability of each differential characteristic is at most $p^{-3} \leq 2^{-1.5\cdot\kappa}$, which we conjecture to be sufficient for preventing differential (and, more generally, other statistical) attacks in our scenario.

8.2 Algebraic and Gröbner Basis Attacks

We start by pointing out that it is not possible to mount an interpolation attack, since the input y, z is unknown, and the polynomials associated with the various heads $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(y, \mathcal{R}_i(z), i)$ differ for each i.

Thus the remainder of this section will be devoted to Gröbner basis attacks. Note that the variables y and z are clearly not independent, as they both depend on x. Moreover, z can be written out as a function of y (the converse does not hold, since the function that outputs z is, in general, not invertible). However, the mentioned functions would be dense and reach maximum degree, which implies that the cost of an attack making use of these functions would be prohibitively expensive (see Section 3.2 for details). Hence, we will treat y and z as independent variables in the following.

Preliminaries: Gröbner Basis Attacks. The most efficient methods for solving multivariate systems over large finite fields involve computing a Gröbner basis associated with the system. We refer the reader to [17] for details on the underlying theory.

Computing a Gröbner basis (in the grevlex order) is, in general, only one of the steps involved in solving a system of polynomials. In our setting, an attacker is able to set up an overdetermined polynomial system where a unique solution can be expected. In this case it is often possible to read the solution directly from the grevlex Gröbner basis, which is why we will solely focus on the step of computing said basis. There are no general complexity estimates for the running time of state-of-the-art Gröbner basis algorithms, such as F$_4$ [27]. There is, however, an important class of polynomial systems, known as semi-regular (see [7] for a definition), that is well understood. For a semi-regular system the degree of the polynomials encountered in F$_4$ is expected to reach the degree of regularity which can, in this case, be defined as the index of the first non-positive coefficient in the series

$$H(z) = \frac{\prod_{i=1}^{n_e}(1 - z^{d_i})}{(1 - z)^{n_v}}, \tag{8}$$

for a system of n_e polynomials in n_v variables, where d_i is the degree of the i-th equation. The time complexity for computing a grevlex Gröbner basis for such a system is then estimated by

$$\mathcal{O}\left(\binom{D_{\mathrm{reg}}+n_v}{n_v}^{\omega}\right), \tag{9}$$

where $2 \leq \omega \leq 3$ is the linear algebra constant representing the cost of matrix multiplication, and D_{reg} the associated degree of regularity [7].

Gröbner Basis Attacks on $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$. There are many possible ways to represent a cipher as a system of multivariate polynomials, and this choice of representation impacts the performance of the Gröbner basis algorithm. We will consider the two representations that seem most promising. The first representation only works with the input and the output of the function (i.e., no additional variables are introduced). This allows an attacker to take advantage of collecting many outputs (each of which will yield a new equation in the same variables), up until the entire polynomial system can be linearized. The details of this first strategy are outlined in Appendix H.2. The second representation introduces new variables and equations for each round. While this increases the number of variables, it keeps the degree low, and allows exploitation of the small number of multiplications in each round. Since the second strategy yields the better attack, here we sum up our findings, with more details included in Appendix H.3.

As described in Appendix H.3, the most promising intermediate modeling can be reduced to a system of R_{H} quadratic equations in $R_{\mathrm{H}} - 2$ variables, where R_{H} is the number of rounds in $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$. Further analysis presented in the same appendix shows that while this system is not semi-regular, we nevertheless find that the degrees encountered in the F_4 algorithm is estimated well by the series $H(z)$ in Eq. (8), and that the solving times are comparable to that of solving randomly generated semi-regular systems of the same size. Still, the systems from $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ have some properties we do not expect in semi-regular systems. To ensure that they cannot be exploited, we conservatively add 3 extra rounds on top of this baseline (see Appendix H.3 for a more detailed exposition of the underlying arguments). Hence, for a security level κ we follow Eq. (9) and define $R^* = R^*(\kappa)$ to be the minimum positive integer that satisfies the following inequality

$$\binom{2 \cdot R^* - 2 + D_{\mathrm{reg}}}{2 \cdot R^* - 2}^2 \geq 2^{\kappa} \quad \text{where} \quad D_{\mathrm{reg}} = R^* + 1 - \left\lfloor \frac{\sqrt{2 \cdot R^* + 2}}{2} \right\rfloor . \tag{10}$$

Appendix H.4 provides details on how this particular expression for D_{reg} is derived from the series in Eq. (8). We then claim that $R^*(\kappa) + 3$ is sufficient to provide κ-bit security against this Gröbner basis attack.

9 Hydra in MPC Applications

In this section, we evaluate the performance of Hydra compared to other PRFs in MPC use cases which assume a secret shared key. We implemented Hydra,

Table 1: Online and offline phase performance for evaluating different ciphers with different state sizes t in MPC using a secret shared key. *Prec* is the number of precomputed elements, i.e., multiplication triples, squares, and inverses; *Depth* describes the number of online communication rounds. Runtime averaged over 200 runs. **Bold** values are best values with key schedules, *Italic* the best without.

		Offline			Online			Combined	
Cipher	Rounds	Prec.	Time ms	Data MB	Depth	Time ms	Data kB	Time ms	Data MB
$t = 8$:									
Hydra	8, 38, 38	**216**	**41.09**	**4.87**	139	**7.99**	7.19	**49.09**	**4.88**
Ciminion (No KS)[a]	90, 14	*148*	*27.30*	*3.34*	107	*4.10*	*5.02*	*31.39*	*3.35*
Ciminion	90, 14	867	181.50	19.55	735	20.47	28.02	201.97	19.58
HadesMiMC	6, 71	238	45.77	5.37	79	13.84	**5.99**	59.62	5.38
Rescue (No KS)[a]	10	480	97.08	10.83	*33*	5.95	11.80	103.03	10.84
Rescue	10	960	205.17	21.65	***33***	10.44	23.32	215.61	21.68
$t = 32$:									
Hydra	8, 38, 38	**330**	**63.69**	**7.44**	139	**12.80**	**11.22**	**76.49**	**7.46**
Ciminion (No KS)[a]	90, 14	*328*	*63.31*	*7.40*	119	*5.38*	*11.16*	*68.69*	*7.41*
Ciminion	90, 14	3207	808.91	72.30	2895	71.92	103.29	880.83	72.41
HadesMiMC	6, 71	526	108.22	11.87	79	162.15	13.29	270.36	11.88
Rescue (No KS)[a]	10	1920	409.76	43.30	*33*	30.94	46.74	440.70	43.35
Rescue	10	3840	983.05	86.60	***33***	77.30	92.82	1060.35	86.70
$t = 64$:									
Hydra	8, 38, 38	***520***	***105.23***	***11.73***	139	**20.20**	***17.82***	**125.43**	***11.75***
Ciminion (No KS)[a]	90, 14	568	117.38	12.81	135	*6.77*	19.35	*124.15*	12.83
Ciminion	90, 14	6327	1819.67	142.64	5775	141.30	203.64	1960.97	142.84
HadesMiMC	6, 71	910	191.50	20.53	79	613.61	23.02	805.11	20.55
Rescue (No KS)[a]	10	3840	989.10	86.60	*33*	97.35	93.34	1086.44	86.70
Rescue	10	7680	2255.42	173.20	***33***	270.98	185.50	2526.40	173.39
$t = 128$:									
Hydra	8, 38, 38	***862***	***178.32***	***19.44***	139	**32.19**	***29.78***	***210.51***	***19.47***
Ciminion (No KS)[a]	90, 14	1048	220.69	23.63	167	*9.96*	35.74	230.65	23.67
Ciminion	90, 14	12567	3842.51	283.32	11535	272.65	404.34	4115.16	283.72
HadesMiMC	6, 71	1678	361.24	37.85	79	2443.52	42.47	2804.76	37.89
Rescue (No KS)[a]	10	7680	2284.69	173.20	*33*	359.10	186.52	2643.79	173.39
Rescue	10	15360	4740.96	346.40	***33***	1022.45	370.84	5763.41	346.77

[a] Assumes round keys are present, i.e., no key schedule computation in MPC.

alongside its competitors, using the MP-SPDZ library [40][14] (version 0.2.8, files can be found in Appendix A) and benchmark it using SPDZ [22, 21] with the MASCOT [41] offline phase protocol. Concretely, we benchmark a two-party setting in a simulated LAN network (1 Gbit/s and $\ll$ 1 ms average round-trip time) using a Xeon E5-2669v4 CPU (2.6 GHz, turboboost up to 3.6 GHz), where each party was assigned only 1 core. SPDZ, and therefore all the PRFs,

[14] `https://github.com/data61/MP-SPDZ/`

is instantiated using a 128-bit prime p, with $\gcd(3, p-1) = \gcd(5, p^2-1) = 1$, thus ensuring that $x \mapsto x^3$ and $x \mapsto \mathcal{D}_{5,\alpha}(x)$ are permutations, as required by POSEIDON, *Rescue*, MiMC, GMiMC, and HYDRA). All PRFs are instantiated with $\kappa = 128$. HYDRA requires $\#\text{MUL} = R_{\mathcal{E}} \cdot (2\left\lfloor \frac{d}{2} \right\rfloor + 4) + 2 \cdot R_{\mathcal{I}} + R_{\text{H}} \cdot \#\text{HEADS}$ multiplications, hence $140 + 38 \cdot \#\text{HEADS}$ in this setting.

We implemented all x^3 evaluations using the technique from [31], which requires one precomputed beaver triple, one precomputed shared random square, and one online communication round. Furthermore, we implemented $x^{1/3}$ (as used in *Rescue*) using the technique described in [5], which requires two precomputed beaver triples, one precomputed random square, one precomputed random inverse, and two communication rounds.[15]

In Table 1, we compare the performance of HYDRA to some competitors for different state sizes t, for a comparison with more PRFs we refer to Appendix I. For Ciminion and *Rescue* we give the performance for the case in which the key schedule has to be computed (i.e., a fresh shared key is provided), and the case in which an old key is used (i.e., round keys can be reused). We give concrete runtimes, as well as the amount of data transmitted by each party during evaluation of the offline and online phases. Further, we give the combined number of triples, squares, and inverses which need to be created during the offline phase, as well es the total number of communication rounds (i.e., the depth of the PRF) during the online phase. During the offline phase only the required number of triples, squares, and inverses is precomputed.

Table 1 shows that the offline phase dominates both the overall runtime and the total communication between the parties. HYDRA always requires less precomputation than HADESMiMC and *Rescue* (with and without a key schedule), hence, it has a significantly more efficient offline phase with the advantage growing with t. Compared to Ciminion one can observe that if one has to compute the key schedule its offline performance is worse than HADESMiMC, consequently also worse than HYDRA. Only if we do not consider the key schedule, Ciminion has a more efficient offline phase than HYDRA for small $t < 64$.

Looking a the online phase, HYDRA is faster and requires less communication than HADESMiMC and *Rescue*, which is due to the smaller number of multiplications and the better plain performance. HADESMiMC requires many expensive MDS matrix multiplications (see Appendix J) and *Rescue* requires expensive $x^{1/d}$ evaluations. Comparing HYDRA to Ciminion without a key schedule, one can observe that Ciminion is always faster. For small state sizes $t < 64$ it also requires less communication. However, Ciminion has an expensive key schedule, which drastically impacts its online phase performance if it has to be computed. Thus, considering key schedules, HYDRA also has the most efficient online phase.

To summarize, our experiments show that HYDRA is the most efficient PRF in both phases of the MPC protocols. Only if we discard the key schedules, Ciminion is competitive for small state sizes $t < 64$.

[15] Precomputed squares and precomputed inverses can be computed in the offline phase using in total one extra round of communication from one beaver triple each.

The Effect of the Network. The performance of any MPC application depends on the network speed. On one hand, the lower the bandwidth, the bigger the effect of the communication between the parties on the overall performance; on the other hand, a big round-trip time leads to bigger contributions of the number of communication rounds. In the offline phase only shared correlated randomness is created, thus, the network performance affects all PRF in the same way. Consequently, if a PRF has a faster offline phase in the LAN setting, it is also faster in a slower network environment. The situation is quite different in the online phase: In very fast networks, the online phase performance is mostly determined by the plain runtime. The slower a network gets, the more time is spent waiting for the network to deliver packages. HYDRA has a small number of multiplication, hence, in all networks a preferable offline phase. Further, it requires very little communication in the online phase, making it suitable for low bandwidth networks. However, it has a larger depth compared to HADESMiMC and *Rescue*, leading to worse runtimes in high-delay networks. Ciminion's key schedule has a large depth and requires lots of communication between the parties. Thus, Ciminion is only competitive in slow networks if the key schedule does not need to be computed. Overall, HYDRA has a good balance between small number of multiplications, small communication, decent plain performance, as well as a reasonable depth, making it the preferred PRF for MPC applications in most network environments.

Acknowledgments. Lorenzo Grassi is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Morten Øygarden has been funded by The Research Council of Norway through the project "qsIoT: Quantum safe cryptography for the Internet of Things". Roman Walch is supported by the "DDAI" COMET Module within the COMET – Competence Centers for Excellent Technologies Programme, funded by the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry for Digital and Economic Affairs (bmdw), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia. The COMET Programme is managed by FFG.

References

[1] D. Abram, I. Damgård, P. Scholl, and S. Trieflinger. "Oblivious TLS via Multiparty Computation". In: *CT-RSA*. Vol. 12704. LNCS. 2021, pp. 51–74.
[2] M. R. Albrecht, L. Grassi, L. Perrin, S. Ramacher, C. Rechberger, D. Rotaru, A. Roy, and M. Schofnegger. "Feistel Structures for MPC, and More". In: *ESORICS*. Vol. 11736. LNCS. 2019, pp. 151–171.
[3] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". In: *ASIACRYPT*. Vol. 10031. LNCS. 2016, pp. 191–219.

[4] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. “Ciphers for MPC and FHE”. In: *EUROCRYPT*. Vol. 9056. LNCS. 2015, pp. 430–454.

[5] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. “Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols”. In: *IACR Transactions on Symmetric Cryptology* 2020.3 (2020), pp. 1–45.

[6] E. Andreeva, V. Lallemand, A. Purnal, R. Reyhanitabar, A. Roy, and D. Vizár. “Forkcipher: A New Primitive for Authenticated Encryption of Very Short Messages”. In: *ASIACRYPT*. Vol. 11922. LNCS. 2019, pp. 153–182.

[7] M. Bardet, J.-C. Faugére, B. Salvy, and B.-Y. Yang. “Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems”. In: *Proc. of MEGA*. Vol. 5. 2005.

[8] D. Beaver. “Efficient Multiparty Protocols Using Circuit Randomization”. In: *CRYPTO*. Vol. 576. LNCS. 1991, pp. 420–432.

[9] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. V. Assche, and R. V. Keer. “Farfalle: parallel permutation-based cryptography”. In: *IACR Trans. Symmetric Cryptol.* 2017.4 (2017), pp. 1–38.

[10] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. “On the Indifferentiability of the Sponge Construction”. In: *EUROCRYPT*. Ed. by N. P. Smart. Vol. 4965. LNCS. 2008, pp. 181–197.

[11] T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. “Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems”. In: *CRYPTO*. Vol. 12172. LNCS. 2020, pp. 299–328.

[12] E. Biham and A. Shamir. “Differential Cryptanalysis of DES-like Cryptosystems”. In: *CRYPTO*. Vol. 537. LNCS. 1990, pp. 2–21.

[13] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[14] P. Bogetoft, I. Damgård, T. P. Jakobsen, K. Nielsen, J. Pagter, and T. Toft. “A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation”. In: *Financial Cryptography*. Vol. 4107. LNCS. 2006, pp. 142–147.

[15] C. Chaigneau, T. Fuhr, H. Gilbert, J. Guo, J. Jean, J.-R. Reinhard, and L. Song. “Key-Recovery Attacks on Full Kravatte”. In: *IACR Trans. Symmetric Cryptol.* 2018.1 (2018), pp. 5–28.

[16] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. “Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives”. In: *CCS*. ACM, 2017, pp. 1825–1842.

[17] D. Cox, J. Little, and D. O’Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.

[18] T. Cui and L. Grassi. “Algebraic Key-Recovery Attacks on Reduced-Round Xoofff”. In: *SAC*. Vol. 12804. LNCS. 2020, pp. 171–197.

[19] J. Daemen. “Limitations of the Even-Mansour Construction”. In: *ASIACRYPT*. Vol. 739. LNCS. 1991, pp. 495–498.

[20] J. Daemen, S. Hoffert, G. V. Assche, and R. V. Keer. “The design of Xoodoo and Xoofff”. In: *IACR Trans. Symmetric Cryptol.* 2018.4 (2018), pp. 1–38.

[21] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. “Practical Covertly Secure MPC for Dishonest Majority - Or: Breaking the SPDZ Limits”. In: *ESORICS*. Vol. 8134. LNCS. 2013, pp. 1–18.

[22] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. “Multiparty Computation from Somewhat Homomorphic Encryption”. In: *CRYPTO*. Vol. 7417. LNCS. Springer, 2012, pp. 643–662.

[23] C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. “Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit”. In: *CRYPTO*. Vol. 10991. LNCS. 2018, pp. 662–692.

[24] C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. “Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields”. In: *EUROCRYPT*. Vol. 12697. LNCS. 2021, pp. 3–34.

[25] O. Dunkelman, N. Keller, and A. Shamir. “Minimalism in Cryptography: The Even-Mansour Scheme Revisited”. In: *EUROCRYPT*. Vol. 7237. LNCS. 2012, pp. 336–354.

[26] S. Even and Y. Mansour. “A Construction of a Cipher From a Single Pseudorandom Permutation”. In: *ASIACRYPT*. Vol. 739. LNCS. 1991, pp. 210–224.

[27] J.-C. Faugére. “A new efficient algorithm for computing Gröbner bases (F_4)”. In: *Journal of pure and applied algebra* 139.1-3 (1999), pp. 61–88.

[28] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. “On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy”. In: *EUROCRYPT*. Vol. 12106. LNCS. 2020, pp. 674–704.

[29] L. Grassi, S. Onofri, M. Pedicini, and L. Sozzi. *Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over $\mathbb{F}_p^n$*. Cryptology ePrint Archive, Report 2021/1695. https://ia.cr/2021/1695. 2021.

[30] L. Grassi, C. Rechberger, and S. Rønjom. “Subspace Trail Cryptanalysis and its Applications to AES”. In: *IACR Trans. Symmetric Cryptol.* 2016.2 (2016), pp. 192–225.

[31] L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart. “MPC-Friendly Symmetric Key Primitives”. In: *CCS*. ACM, 2016, pp. 430–443.

[32] L. Grassi, C. Rechberger, and M. Schofnegger. “Proving Resistance Against Infinitely Long Subspace Trails: How to Choose the Linear Layer”. In: *IACR Trans. Symmetric Cryptol.* 2021.2 (2021), pp. 314–352.

[33] A. Gunsing and B. Mennink. “The Summation-Truncation Hybrid: Reusing Discarded Bits for Free”. In: *CRYPTO*. Vol. 12170. LNCS. 2020, pp. 187–217.

[34] C. Guo, F.-X. Standaert, W. Wang, X. Wang, and Y. Yu. “Provable Security of SP Networks with Partial Non-Linear Layers”. In: *IACR Transactions on Symmetric Cryptology* 2021.2 (2021), pp. 353–388.

[35] C. Hall, D. A. Wagner, J. Kelsey, and B. Schneier. “Building PRFs from PRPs”. In: *CRYPTO*. Vol. 1462. LNCS. 1998, pp. 370–389.

[36] L. Helminger, D. Kales, S. Ramacher, and R. Walch. “Multi-party Revocation in Sovrin: Performance through Distributed Trust”. In: *CT-RSA*. Vol. 12704. LNCS. Springer, 2021, pp. 527–551.

[37] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. “Zero-knowledge from secure multiparty computation”. In: *STOC*. ACM, 2007, pp. 21–30.

[38] T. Jakobsen and L. R. Knudsen. “The Interpolation Attack on Block Ciphers”. In: *FSE*. Vol. 1267. LNCS. 1997, pp. 28–40.

[39] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert. “Mobile Private Contact Discovery at Scale”. In: *USENIX Security Symposium*. USENIX Association, 2019, pp. 1447–1464.

[40] M. Keller. “MP-SPDZ: A Versatile Framework for Multi-Party Computation”. In: *CCS*. ACM, 2020, pp. 1575–1590.

[41] M. Keller, E. Orsini, and P. Scholl. “MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer”. In: *CCS*. ACM, 2016, pp. 830–842.

[42] L. R. Knudsen. “Truncated and Higher Order Differentials”. In: *FSE*. Vol. 1008. LNCS. 1994, pp. 196–211.

[43] X. Lai and J. L. Massey. “A Proposal for a New Block Encryption Standard”. In: *EUROCRYPT*. Vol. 473. LNCS. 1990, pp. 389–404.

[44] S. Laur, R. Talviste, and J. Willemson. “From Oblivious AES to Efficient and Secure Database Join in the Multiparty Setting”. In: *ACNS*. Vol. 7954. LNCS. 2013, pp. 84–101.

[45] G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. “A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack”. In: *CRYPTO*. Vol. 6841. LNCS. 2011, pp. 206–221.

[46] G. Leander, B. Minaud, and S. Rønjom. “A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro”. In: *EUROCRYPT*. Vol. 9056. LNCS. 2015, pp. 254–283.

[47] S. Lucks. “The Sum of PRPs Is a Secure PRF”. In: *EUROCRYPT*. Vol. 1807. LNCS. 2000, pp. 470–484.

[48] B. Mennink and S. Neves. “Optimal PRFs from Blockcipher Designs”. In: *IACR Transactions on Symmetric Cryptology* 2017.3 (2017), pp. 228–252.

[49] P. Mohassel and Y. Zhang. “SecureML: A System for Scalable Privacy-Preserving Machine Learning”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017, pp. 19–38.

[50] G. L. Mullen and D. Panario. *Handbook of Finite Fields*. 1st. Chapman & Hall/CRC, 2013.

[51] A. Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (1979), pp. 612–613.

[52] S. Vaudenay. “On the Lai-Massey Scheme”. In: *ASIACRYPT*. Vol. 1716. LNCS. 1999, pp. 8–19.

SUPPLEMENTARY MATERIAL

A Supplementary Files

In the repository

https://extgit.iaik.tugraz.at/krypto/hydra

we provide the following files as supplementary material.

- `MP-SPDZ`: This folder contains the MP-SPDZ framework and cipher implementations used for benchmarking.
- `calc_round_numbers.sage`: This script calculates the number of rounds for a given security level κ.
- `hydra.sage`: This script contains the reference implementation of Hydra, written in sage.

We refer to the `Readme.md` file in the repository for more information about the supplied files.

B Details about Farfalle

Let $\mathtt{K} \in \mathbb{F}_q^\kappa$ be the secret key for $\kappa \geq 1$. Farfalle uses six components, namely

1. key schedules $\mathcal{K}^c : \mathbb{F}_q^\kappa \to \mathbb{F}_q^\star$ and $\mathcal{K}^e : \mathbb{F}_q^\kappa \to \mathbb{F}_q^\star$ for generating the subkeys used in the compression and in the expansion phases respectively,[16] that is,
$$\mathcal{K}^c(\mathtt{K}) = (k_0^{(c)}, k_1^{(c)}, k_2^{(c)}, \ldots), \qquad \mathcal{K}^e(\mathtt{K}) = (k_0^{(e)}, k_1^{(e)}, k_2^{(e)}, \ldots),$$
2. two keyed functions $\mathcal{F}_k^{(c)}, \mathcal{F}_k^{(e)} : \mathbb{F}_q \to \mathbb{F}_q$ defined via two unkeyed functions $\mathcal{F}^{(c)}, \mathcal{F}^{(e)} : \mathbb{F}_q \to \mathbb{F}_q$ as
$$\mathcal{F}_k^{(c)}(x) = \mathcal{F}^{(c)}(k + x), \quad \mathcal{F}_k^{(e)}(x) = k + \mathcal{F}^{(e)}(x),$$
3. an unkeyed function $\mathcal{F}' : \mathbb{F}_q \to \mathbb{F}_q$,
4. a compression function $\mathcal{C} : \mathbb{F}_q^\star \to \mathbb{F}_q$, defined in [2] as $\mathcal{C}(x_0, x_1, \ldots, x_{n_c-1}) = \sum_{j=0}^{n_c-1} x_j$, and
5. an expansion function $\mathcal{E} : \mathbb{F}_q \to \mathbb{F}_q^\star$ defined as
$$\mathcal{E}(x) = (\mathcal{E}_0(x), \mathcal{E}_1(x), \mathcal{E}_2(x), \ldots) \tag{11}$$
for some functions $\mathcal{E}_i : \mathbb{F}_q \to \mathbb{F}_q$.

Given an input $x = (x_0, \ldots, x_{n_c-1}) \in \mathbb{F}_q^{n_c}$, Farfalle : $\mathbb{F}_q^{n_c} \to \mathbb{F}_q^{n_e}$ is defined by

$$\text{Farfalle}(x) = \underbrace{(\mathcal{F}^{(e)}_{k_0^{(e)}}, \mathcal{F}^{(e)}_{k_1^{(e)}}, \cdots, \mathcal{F}^{(e)}_{k_{n_e-1}^{(e)}})}_{n_e \text{ times}} \circ \mathcal{E} \circ \mathcal{F}' \left(\sum_{j=0}^{n_c-1} \mathcal{F}^{(c)}_{k_j^{(c)}}(x_j) \right),$$

where $(\mathcal{F}^{(e)}_{k_0^{(e)}}, \cdots, \mathcal{F}^{(e)}_{k_{n_e-1}^{(e)}})(z) := \big(\mathcal{F}^{(e)}_{k_0^{(e)}}(z_0), \cdots, \mathcal{F}^{(e)}_{k_{n_e-1}^{(e)}}(z_{n_e-1})\big)$ for any $z = (z_0, z_1, \ldots, z_{n_e-1}) \in \mathbb{F}_q^{n_e}$.

[16] In [2], given the master key $\mathtt{K}$, an initial mask K is defined as $K := \mathcal{P}^{(b)}(\mathtt{K} \mid\mid 10^\star)$. The masks $k_i^{(c)}$ and $k_j^{(e)}$ are obtained via a rolling function applied on K.

C The Hydra PRF – Specification Details

C.1 Pseudo Code

Algorithm 1: The Hydra PRF.

```
Data: Prime integer $p \geq 2^{63}$, $4 \leq t \leq 2^{\kappa/2}$, $N \in \mathbb{F}_p$, $\mathtt{K} \in \mathbb{F}_p^4$.
Result: $h \in \mathbb{F}_p^t$.
1  Let $t = 14 \cdot t' + t''$ for $t', t'' \in \mathbb{N}$, where $t'' = t \mod 14$.
2  Let $x, y, z \leftarrow 0 \in \mathbb{F}_p^4$.
   // First step (computing $\mathcal{P}^{(\mathrm{B})}$)
3  $x \leftarrow M_{\mathcal{E}} \times (\mathtt{K} + [N \,||\, \mathtt{IV}])$.
4  for $i \leftarrow 0$ to $3$ do
5      $x \leftarrow \mathcal{E}_i(x)$.
6      $z \leftarrow z + x$.
7  for $i \leftarrow 0$ to $R - 1$ do
8      $x \leftarrow \mathcal{I}_i(x)$.
9      $z \leftarrow z + x$.
10 for $i \leftarrow 4$ to $6$ do
11     $x \leftarrow \mathcal{E}_i(x)$.
12     $z \leftarrow z + x$.
13 $y \leftarrow \mathcal{E}_7(x) + \mathtt{K}$.
   // Expansion step (using $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$)
14 for $i \leftarrow 0$ to $t' - 1$ do
15     $a \leftarrow \mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(y, \mathcal{R}(z, 2 \cdot i), 2 \cdot i)$.
16     $b \leftarrow \mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(y, \mathcal{R}(z, 2 \cdot i + 1), 2 \cdot i + 1)$.
17     $h_i \leftarrow \mathcal{T}_{8,6}^{L}(a) \,||\, (\mathcal{T}_{8,2}^{R}(a) + \mathcal{T}_{8,2}^{L}(b)) \,||\, \mathcal{T}_{8,6}^{R}(b)$.
18 $h_{t'} \leftarrow 0$.
19 if $t'' > 0$ then
20     $a \leftarrow \mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(y, \mathcal{R}(z, 2 \cdot t'), 2 \cdot t')$.
21     $h_{t'} \leftarrow \mathcal{T}_{8,\min\{t'',6\}}^{L}(a)$.
22     if $t'' > 6$ then
23         $b \leftarrow \mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(y, \mathcal{R}(z, 2 \cdot t' + 1), 2 \cdot t' + 1)$.
24         $h_{t'} \leftarrow \mathcal{T}_{14,t''}^{L}\left(h_{t'} \,||\, (\mathcal{T}_{8,2}^{R}(a) + \mathcal{T}_{8,2}^{L}(b)) \,||\, \mathcal{T}_{8,6}^{R}(b)\right)$.
25 return $h := h_0 \,||\, h_1 \,||\, \cdots \,||\, h_{t'-1} \,||\, h_{t'} \in \mathbb{F}_p^t$.
```

C.2 Generation of Matrices and Constants

We pseudo-randomly generate all matrices and constants using Shake-128 [3] seeded with the string Hydra and the used prime. Thereby, we use rejection sampling to sample field elements, and we reject and resample matrices and constants if they do not meet the requirements specified in this paper. We refer to our source code present in the supplementary material (Appendix A) for more details.

D Statistical Attacks against $\mathcal{P}^{(\mathrm{B})}$

Similar to HadesMiMC and Poseidon, we make use of the external rounds in order to provide security against statistical attacks. As already discussed before, we claim that the security against statistical attacks of $\mathcal{P}^{(\mathrm{B})}$ is not lower than that of

$$\widehat{\mathcal{P}^{(\mathrm{B})}}(\cdot) := \underbrace{\mathcal{F} \circ \mathcal{F}}_{2 \text{ times}} \circ L_{\mathcal{P}^{(\mathrm{B})}} \circ \underbrace{\mathcal{F} \circ \cdots \circ \mathcal{F}}_{4 \text{ times}}(\cdot)\,.$$

For this reason, here we study the security of $\widehat{\mathcal{P}^{(\mathrm{B})}}$ against statistical attacks.

D.1 Differential Cryptanalysis

Differential cryptanalysis [6, 7] and its variations are the most widely used techniques to analyze symmetric-key primitives. Given pairs of inputs with fixed input differences, differential attacks consider the probability distribution of the corresponding output differences produced by the cryptographic primitive. Let $\delta_I, \delta_O \in \mathbb{F}_p^t$ be respectively the input and the output differences through a function f over $\mathbb{F}_p^t$. The differential probability (DP) of having a certain output difference δ_O given a particular input difference δ_I is equal to

$$\mathrm{Prob}(\delta_I \to \delta_O) = \frac{|\{x \in \mathbb{F}_p^t \mid f(x + \Delta_I) - f(x) = \delta_O\}|}{p^t}.$$

As Hydra is an iterated scheme and assuming the independence of the rounds, the DP of a differential trail is the product of the DPs of its one-round differences.

Our goal is to find the minimum number of rounds such that the probability of each differential characteristic is smaller than $2^{-2.5\cdot\kappa}$. Since multiple characteristics can be used simultaneously in order to set up a differential attack, we arbitrary chose the value $2^{-2.5\cdot\kappa}$ instead of $2^{-\kappa}$ to provide security.

Proposition 2. *Let $\alpha \in \mathbb{F}_p \setminus \{0\}$, and let $d \geq 5$ such that $\gcd(d, p^2 - 1) = 1$. Let $\beta \in \mathbb{F}_p \setminus \{0\}$, and let $\widehat{S_{\mathcal{E}}} : \mathbb{F}_p^2 \to \mathbb{F}_p^2$ be defined as*

$$\widehat{S_{\mathcal{E}}}(x_0, x_1) = x_0 \cdot \mathcal{D}'_{d,\alpha}(x_0 + \beta \cdot x_1) \| x_1 \cdot \mathcal{D}'_{d,\alpha}(x_0 + \beta \cdot x_1),$$

where $\mathcal{D}'_{d,\alpha}(\cdot)$ is defined as in Eq. (5). Given an input difference $\Delta_I = (\delta_0, \delta_1) \in \mathbb{F}_p^2 \setminus \{(0,0)\}$ and an output difference $\Delta_O = (\delta'_0, \delta'_1) \in \mathbb{F}_p^2$, we have that

$$\mathit{Prob}(\Delta_I \to \Delta_O) \leq \begin{cases} \frac{d-1}{p} & \text{if } \delta_0 + \beta \cdot \delta_1 = \delta'_0 + \beta \cdot \delta'_1 = 0\,, \\ \frac{(d-1)^2}{p^2} & \text{otherwise.} \end{cases}$$

Proof. Our goal is to count the number of solutions of

$$(x_0 + \delta_0) \cdot \mathcal{D}'_{d,\alpha}(x_0 + \beta \cdot x_1 + \delta_0 + \beta \cdot \delta_1) - x_0 \cdot \mathcal{D}'_{d,\alpha}(x_0 + \beta \cdot x_1) = \delta'_0\,,$$
$$(x_1 + \delta_1) \cdot \mathcal{D}'_{d,\alpha}(x_0 + \beta \cdot x_1 + \delta_0 + \beta \cdot \delta_1) - x_1 \cdot \mathcal{D}'_{d,\alpha}(x_0 + \beta \cdot x_1) = \delta'_1\,.$$

By summing the first with the second multiplied by β, we get

$$\mathcal{D}_{d,\alpha}(x_0 + \beta \cdot x_1 + \delta_0 + \beta \cdot \delta_1) - \mathcal{D}_{d,\alpha}(x_0 + \beta \cdot x_1) = \delta'_0 + \beta \cdot \delta'_1 \,,$$

where $\mathcal{D}_{d,\alpha}(x) := x \cdot \mathcal{D}'_{d,\alpha}(x)$ for each $x \in \mathbb{F}_p$.

Let's introduce the variable z:

$$z := x_0 + \beta \cdot x_1 \,.$$

The system of equations becomes

$$\begin{aligned}(x_0 + \delta_0) \cdot \mathcal{D}'_{d,\alpha}(z + \delta_0 + \beta \cdot \delta_1) - x_0 \cdot \mathcal{D}'_{d,\alpha}(z) &= \delta'_0 \,,\\ \mathcal{D}_{d,\alpha}(z + \delta_0 + \beta \cdot \delta_1) - \mathcal{D}_{d,\alpha}(z) &= \delta'_0 + \beta \cdot \delta'_1 \,.\end{aligned}$$

As a result, the second equation depends only on z, which is independent of x_0 (since x_1 is independent of x_0).

Note that if

$$\delta_0 + \beta \cdot \delta_1 = \delta'_0 + \beta \cdot \delta'_1 = 0 \,,$$

then the second equation is always satisfied, independently of z. Otherwise, it admits at most $d - 1$ solutions (since it is an equation of degree $d - 1$ in z).

For each solution z of the second equation, the first equation admits at most $d - 1$ solutions (since again it is an equation of degree $d - 1$ in x_0). □

It follows that the *maximum differential probability* of $S_{\mathcal{E}}^{(\cdot)}$ over $\mathbb{F}_p^2$ is at most $(d - 1)/p < d/p$, that is,

$$\max_{\Delta_I, \Delta_O \in \mathbb{F}_p^2 \setminus \{(0,0)\}} \text{Prob}\left(\Delta_I \xrightarrow{S_{\mathcal{E}}^{(\cdot)}} \Delta_O\right) \leq \frac{d}{p} \,.$$

The wide-trail design strategy [13] does not apply directly to $\widehat{\mathcal{P}^{(\mathrm{B})}}$, since $S_{\mathcal{E}}^{(\cdot)}$ mixes two $\mathbb{F}_p$ elements in a nonlinear way and since $M_{\mathcal{E}}$ is defined as a matrix multiplication in $\mathbb{F}_p^{4\times 4}$. However,

(1) since $M_{\mathcal{E}}$ is an MDS matrix, at least five $\mathbb{F}_p$ elements must be active among the output of $\mathcal{S}_{\mathcal{E}}$ and the input of the consecutive $\mathcal{S}_{\mathcal{E}}$, and

(2) since $\mathcal{S}_{\mathcal{E}}^{(\cdot)}$ is defined over $\mathbb{F}_p^2$, at least three $\mathcal{S}_{\mathcal{E}}^{(\cdot)}$ functions are active over two rounds.

It follows that the probability of any differential characteristic over two rounds is at most $(d/p)^3$, which implies that for three pairs of consecutive rounds it is

$$\frac{d^9}{p^9} \leq \frac{d^9}{2^{4.5\cdot\kappa}} \ll 2^{-2.5\cdot\kappa},$$

since $p^{-2} \leq 2^{-\kappa}$ and $5 \leq d \ll p$.

For completeness, we mention that the probability of a differential characteristic for $\mathcal{P}^{(\mathrm{B})}$ is even smaller. Indeed, following [21], it corresponds to

$$\frac{d^9}{p^9} \cdot \left(\frac{3}{p}\right)^{\left\lfloor \frac{R_{\mathcal{I}}}{4} \right\rfloor},$$

where at least one quadratic Lai–Massey function is active every four rounds (due to the considerations made in Appendix F.1) and the probability of a differential for one degree-$\hat{d}$ Lai–Massey function (where $\hat{d} = 4$ in our case) is at most $(\hat{d} - 1)/p$ (see Lemma 2 for more details).

D.2 Truncated Differential Attacks

Truncated differential cryptanalysis [22] is a variant of classical differential cryptanalysis, in which the attacker can specify only part of the difference between pairs of texts. As shown in [26], truncated differential and subspace trails are strictly related.

Definition 3 ((Invariant) Subspace Trail [24, 25, 17]). *Let $\mathcal{I} : \mathbb{F}_p^t \to \mathbb{F}_p^t$. Let $(\mathfrak{U}_0, \ldots, \mathfrak{U}_r)$ denote a set of $r+1$ subspaces with $\dim(\mathfrak{U}_i) \leq \dim(\mathfrak{U}_{i+1})$. If for each $i \in \{1, \ldots, r\}$ and for each $a_i \in \mathbb{F}^t$ there exists $a_{i+1} \in \mathbb{F}^t$ such that $\mathcal{I}(\mathfrak{U}_i + a_i) \subseteq \mathfrak{U}_{i+1} + a_{i+1}$, then $(\mathfrak{U}_0, \ldots, \mathfrak{U}_r)$ is a subspace trail of length r. If the subspace is invariant (that is, $\mathfrak{U}_i = \mathfrak{U}_j$ for each $i, j = 0, \ldots, r$), the trail is called an* invariant subspace trail*.*

The linear layer $M_{\mathcal{E}} \in \mathbb{F}_p^{4\times 4}$ of $\mathcal{E}$ must be chosen such that (1st) it is MDS and (2nd) no invariant and iterative subspace trails generated by $\langle [1, -1, 0, 0] \rangle \subseteq \mathbb{F}_p^4$ or $\langle [0, 0, 1, 1] \rangle \subseteq \mathbb{F}_p^4$ exist. Indeed, note that the subspaces generated by $\langle [1, -1, 0, 0] \rangle$ and $\langle [0, 0, 1, 1] \rangle$ are invariant through the S-box layer due to the definition of $S_{\mathcal{E}}^{(1)}$ and $S_{\mathcal{E}}^{(2)}$, respectively.

Since we only focus on a state size of 4 elements, our approach is exhaustive. In particular, we first generate all d-dimensional subspaces $\mathfrak{T} \subseteq \mathbb{F}_p^4$, where $d < 4$. For every subspace $\mathfrak{T}$ for which $(1, -1, 0, 0) \in \mathfrak{T}$ and/or $(0, 0, 1, 1) \in \mathfrak{T}$ (i.e., if the subspaces generated by $(1, -1, 0, 0)$ and/or $(0, 0, 1, 1)$ are in $\mathfrak{T}$) we use the following approach.

1. We determine whether $\mathcal{T} = M \times \mathfrak{T}$. If this is the case, we discard the matrix.
2. Otherwise, if $\mathfrak{T} \neq M \times \mathfrak{T}$, we determine if $\mathfrak{T} = M^2 \times \mathfrak{T}$. For example, this happens for matrices which merely move the subspaces from one half of the state to the other (e.g., $M = \mathrm{circ}(0, 0, 1, 0)$), which then results in an iterative subspace trail with $\mathfrak{T} = M^2 \times \mathfrak{T} \neq M \times \mathfrak{T}$. If this is the case, we discard the matrix.

In all other cases, we consider the matrix as viable for our goal. It turns out that the AES matrix $M_{\mathcal{E}} = \mathrm{circ}(3, 2, 1, 1)$ with its fast plain performance fulfills these requirements and can be used in our setting.

By choosing the matrix in such a way, a truncated differential with probability 1 can cover for one round only. Over more rounds, any subspace trail has probability smaller than the security level. It follows that $\widehat{\mathcal{P}^{(\mathrm{B})}}$ (and so $\mathcal{P}^{(\mathrm{B})}$) is secure against truncated differential (equivalently, subspace trail) attacks.

D.3 Other Statistical Attacks

Finally, since $\widehat{\mathcal{P}^{(\mathrm{B})}}$ (and so $\mathcal{P}^{(\mathrm{B})}$) is secure against classical and truncated differential attacks, then we conjecture that it is also secure against other statistical attacks, including linear and zero-correlation attacks [27, 1, 8], impossible differential [5], boomerang attack [29], integral [12], multiple-of-n and mixture differential attacks [18, 15]. This conclusion is also supported by the facts that (1st) full diffusion is achieved over a single round and (2nd) the arrangement of $S_{\mathcal{E}}$ and the one of $M_{\mathcal{E}}$ are different, that is, the nonlinear layer $S_{\mathcal{E}}$ mixes two $\mathbb{F}_p$-elements in a nonlinear way (hence, it admits a natural description over $\mathbb{F}_{p^2}^2$), while $M_{\mathcal{E}}$ is defined as a matrix multiplication in $\mathbb{F}_p^{4\times 4}$, and it does not admit any equivalent representation as a matrix in $\mathbb{F}_{p^2}^{2\times 2}$ (see also [10, 9, 16] for details).

E Details about Algebraic Attacks against $\mathcal{P}^{(\mathrm{B})}$

E.1 Density of the Algebraic Representation

Algebraic attacks are especially efficient against schemes that have a simple algebraic structure in larger fields. It is well-known that many attacks (e.g., the interpolation one or the Gröbner basis one) can be stronger if the polynomials that represent the scheme are sparse. Consequently, it is important to study the density of these polynomials. We recall (see, e.g, Theorem 2.4 in [14]) that the number of possible monomials in a polynomial of degree D in t variables is

$$N(d,t) := \binom{t+D}{D}. \tag{12}$$

Conditions (i) – (ii) on $M_{\mathcal{I}}$. First of all, here we explain the condition *(i) – (ii)* on $M_{\mathcal{I}}$. Let $x \in \mathbb{F}_p^t$ be the input of one round. By simple computation, the j-th $\mathbb{F}_p$-output of the next round is

$$y_j = \sum_{l=0}^{t-1} M_{\mathcal{I}}[j,l] \cdot x_l + \left(\sum_{l=0}^{t-1} M_{\mathcal{I}}[j,l]\right) \cdot G\left(\sum_{h=0}^{t-1} \lambda_h \cdot x_h\right)$$

for a certain (simplified) quadratic function G, and where we omitted constant additions for simplicity. Let $\hat{y} = G\left(\sum_{h=0}^{t-1} \lambda_h \cdot x_h\right)$. By applying the nonlinear

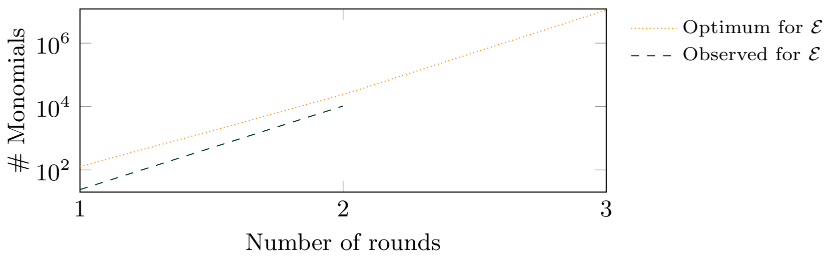


Fig. 4: Comparison of the maximum number of monomials and the observed number of monomials in $\mathcal{P}^{(\mathrm{B})}$ for the $\mathcal{E}$ rounds.

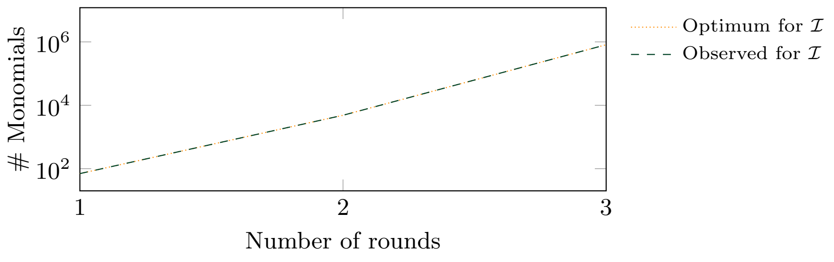


Fig. 5: Comparison of the maximum number of monomials and the observed number of monomials in $\mathcal{P}^{(\mathrm{B})}$ for the $\mathcal{I}$ rounds.

$S_{\mathcal{I}}$, the i-th $\mathbb{F}_p$-output is

$$\begin{aligned}
&y_i + G\left(\sum_{h=0}^{t-1} \lambda_h \cdot \left(\sum_{l=0}^{t-1} M_{\mathcal{I}}[h,l] \cdot x_l + \left(\sum_{l=0}^{t-1} M_{\mathcal{I}}[h,l]\right) \cdot \hat{y}\right)\right) \\
=&y_i + G\left(\sum_{l=0}^{t-1} x_l \cdot \left(\sum_{h=0}^{t-1} \lambda_h \cdot M_{\mathcal{I}}[h,l]\right) + \hat{y} \cdot \sum_{h=0}^{t-1} \lambda_h \cdot \left(\sum_{l=0}^{t-1} M_{\mathcal{I}}[h,l]\right)\right) .
\end{aligned}$$

Hence, the condition $\sum_{h=0}^{t-1} \lambda_h \cdot \left(\sum_{l=0}^{t-1} M_{\mathcal{I}}[h,l]\right) \neq 0$ is crucial in order to ensure the growth of the degree, while the conditions $\sum_{h=0}^{t-1} \lambda_h \cdot M_{\mathcal{I}}[h,l] \neq 0$ for each $l \in \{0,1,\ldots,t-1\}$ ensure that the polynomial is not sparse.

Forward Direction. For the external rounds $\mathcal{E}$, our practical tests show that the maximum number of possible monomials at the i-th round is reached and exceeded in round $i+1$. In the internal rounds $\mathcal{I}$, the growth of the number of monomials is closer to the optimum, reaching its maximum in some of our tests. The behavior of $\mathcal{E}$ and $\mathcal{I}$ is shown in Figs. 4 and 5.

Backward Direction. The inverse of the $S_{\mathcal{E}}^{(\cdot)}$ layer is defined as

$$S_{\mathcal{E}}^{(\cdot)}(y_0, y_1) = \frac{y_0}{\mathcal{D}'_{d,\alpha}\left(\mathcal{D}_{d,\alpha}^{-1}(y_0 \pm y_1)\right)} \,\Big\|\, \frac{y_1}{\mathcal{D}'_{d,\alpha}\left(\mathcal{D}_{d,\alpha}^{-1}(y_0 \pm y_1)\right)} ,$$

as given in the proof of Theorem 1. The inverse of the degree-d Dickson polynomial in $\mathcal{E}$ is a degree-$\hat{d}$ Dickson polynomial, where $d \cdot \hat{d} = 1 \mod p^2 - 1$ (see Theorem 2). Since $d \ll p^2 - 1$, it follows that $\hat{d}$ is of the same order as p, and therefore the inverse has a significantly higher degree and contains more monomials. The same is true for the inverse of $S_{\mathcal{E}}^{(\cdot)}$, as confirmed by our tests. In the $\mathcal{I}$ rounds, the degree is the same in both directions, and the above result is also valid here.

E.2 Interpolation Attack

The goal of the interpolation attack [20] is to construct an interpolation polynomial that describes the function. The cost of setting up such an attack depends on the number of different monomials in the interpolation polynomial, where (an upper/lower bound of) the number of different monomials can be estimated via the degree of the function. If the number of unknown monomials is sufficiently large, then it is not possible to construct the interpolation polynomial faster than via a brute-force attack. Roughly speaking, if the degree of the interpolation polynomial is high enough and if such polynomial is dense/full, then this attack does not work.

Let us consider a scenario in which only one input component is active and the others are fixed, that is, in the case in which the polynomial depends only on a single variable. First of all, note that three external rounds are sufficient to reach the maximum degree in the backward direction. Moreover, note that the output of the first round does not correspond to a dense polynomial of degree d (see Appendix E.1 for more details). Hence, out of six external rounds, only two play a role concerning the number of monomials. Since the data available for constructing the polynomial is restricted to $2^{\kappa/2}$, $R_{\mathcal{I}}$ must satisfy

$$4^{R_{\mathcal{I}}-3} \cdot d^2 \geq 2^{\kappa/2} \implies R_{\mathcal{I}} \geq \frac{\kappa}{4} - \log_2(d) + 3 \ ,$$

where 4 and $d \geq 5$ are the degrees of the internal and of the external rounds respectively, and where we arbitrarily add 3 internal rounds for destroying possible relations existing between the coefficients of the monomials (due to the fact that the degree-4 function that defines the nonlinear layer is not generic, but has a particular structure). As a result,

$$R_{\mathcal{I}} \geq \frac{\kappa}{4} - \log_2(d) + 3 \tag{13}$$

rounds are necessary for preventing MitM interpolation attacks.

E.3 Linearization Attack

Preliminary: Linearization Attacks. Many well-known techniques for solving multivariate polynomial systems of equations, uses linearization (see, e.g., [11]). Given a system of polynomial equations, the idea is to turn it into a system of linear equations by adding new variables that replace all the monomials of the system whose degree is strictly greater than 1. The resulting linear system of equations can be solved using linear algebra if there are enough equations to make the linearized system over-determined, or at least on the same order as the number of variables after linearization.

Consider a system in t unknowns of degree limited by D that we want to linearize. As we have already seen in Eq. (12), the set of monomials considered has cardinality $N(D,t) := \binom{t+D}{D}$. The costs of the attack are given by:

– computational cost of $\mathcal{O}(N(D,t)^\omega)$ operations (for $2 < \omega \leq 3$);
– memory cost of $\mathcal{O}(N(D,t)^2)$ to store the linear equations.

Depending on parameter choices, the hybrid approach which combines exhaustive search with this approach may lead to a reduced cost. Guessing $x < t$ variables leads to a complexity of

$$\mathcal{O}\left(p^x \cdot N(D, t-x)^\omega\right).$$

Linearization Attack on $\mathcal{P}^{(\mathrm{B})}$. For simplicity, we start by assuming that the attacker can collect enough inputs/output data to directly linearize a polynomial system representing $\mathcal{P}^{(\mathrm{B})}$ with 4 key variables. The degree D is given by $D = d^2 \cdot 4^{R_\mathcal{I}-3}$, where (1st) the last three $\mathcal{E}$ rounds are excluded to account for possible MitM variants, and (2nd) one extra external round and three extra $\mathcal{I}$ rounds are required in order to achieve full diffusion (see Appendix E.1 for more details). Since $2^\kappa \leq p^2$, it follows that the attack is not feasible if

$$\forall x \in \{0,1\}: \qquad p^x \cdot \left(\binom{4-x+d^2\cdot 4^{R_\mathcal{I}-3}}{4-x}\right)^2 \geq 2^\kappa,$$

where we consider $\omega = 2$ and where we have 4 variables (namely, the key). Let $x' = 4 - x$. Note that

$$p^x \cdot \left(\binom{D+x'}{x'}\right)^2 \geq p^x \cdot \left(\frac{\prod_{j=1}^{x'}(D+j)}{x'!}\right)^2 \geq p^x \cdot \left(\frac{(D+1)^{x'}}{x'^{x'}}\right)^2 \geq p^x \cdot \left(\frac{D}{x'}\right)^{2\cdot x'}$$

where $z^z \geq z!$ for each $z \geq 1$. Hence, this attack is prevented if

$$R_\mathcal{I} \geq 4 - \log_2(d) + \max_{x\in\{0,1\}} \frac{\kappa - x\cdot \log_2(p)}{4\cdot(4-x)}. \tag{14}$$

Recalling the restriction $4^{R_\mathcal{I}-3}\cdot d^2 \geq 2^{\kappa/2}$ from Appendix E.2, we note that it will, in practice, not be possible to linearize directly, as assumed above. Rather, algorithms such as [11] must go to a higher degree, and linearize a larger system, making the lower bound in Eq. (14) extremely conservative for preventing attacks of this kind. Even when ignoring the issues of data complexity, we point out that the number of rounds necessary for preventing interpolation attacks are largely sufficient for preventing linearization strategies when $\kappa = \log_2(p)$.

E.4 Gröbner Basis Attack

Given a system of n_e nonlinear equations in n_v variables, a Gröbner basis allows to factorize this system of equations and find a solution (if it exists). We refer to Section 8.2 for the details regarding such attack. Here we limit ourselves to recall that

- the cost of such attack depends on the number of nonlinear equations that compose the system to solve, their degree and on the number of variables;
- the cost of the Gröbner basis attack depends on the particular representation of the studied system of equations considered.

Here, we consider the two extreme cases, one in which the attacker only works with the input and the output of the permutation (i.e., no additional variables are introduced), and one in which the attacker introduced intermediate variables in every round.

Inputs and Outputs. The polynomial system in this modeling is the same as described in Appendix E.3, i.e., 4 key variables with polynomials of degree $d^5 \cdot 4^{R_{\mathcal{I}}-3}$. The tools presented in Section 8.2 can be used for estimating how this attack performs, depending on the number of equations (i.e., input/output pairs) the attacker has access to. Similar to what we described in Appendix E.3, this attack is optimized when it is possible to directly linearize the system, meaning that Eq. (14) provides a conservative lower bound on $R_{\mathcal{I}}$ to prevent this strategy as well.

Intermediate Variables. For this strategy, we introduce additional variables in each round in order to reduce the overall degree growth. For simplicity, we focus on a slightly modified permutation with only two external rounds, one at the beginning and one at the end. We emphasize that adding the remaining external rounds will not make the resulting equation system easier to solve.

In our representation, we replace both sums in the Lai–Massey construction of each internal round by two new variables. More formally, let

$$z_i^{(0)} = \sum_{j=0}^{3} \lambda_j^{(0)} \cdot x_j, \quad z_i^{(1)} = \sum_{j=0}^{3} \lambda_j^{(1)} \cdot x_j,$$

where $z_i^{(0)}, z_i^{(1)}$ are two new variables introduced in the i-th internal round. Using this approach, each internal round adds two degree-2 equations. Moreover, we use four variables for the key, and we introduce a layer of four new variables after the first external round (i.e., before starting with the internal rounds in our scenario) and after the last internal round (i.e., before the last external round in our scenario). Note that for the first internal round we can reuse the variables introduced after the first external round, and thus we do not need to add two new variables.

In total, we have thus 4 variables for the key, $2 \cdot 4 = 8$ variables for the transitions between the external and internal rounds, and $2(r-1)$ variables for r internal rounds. The number of equations is the same, and hence

$$n_e = n_v = 12 + 2(r-1).$$

Of these equations, 8 are of degree $d \geq 5$ and $4 + 2(r-1)$ are of degree 4.

We implemented this modeling for an increasing number of r intermediate rounds, and computed the associated grevlex Gröbner basis using the F_4 algorithm on the same setup described in Appendix H.3. For $r = 1$, the basis was computed after almost 4 minutes. When $r = 2$, the same computation took 3 hours and 6 minutes. The case $r = 3$ was manually canceled after over 10 days of running time.

Based on these results, we find it highly unlikely that this approach will outperform the other methods we have investigated. For a rough comparison, consider the interpolation attack in Appendix E.2, which requires $r \gtrapprox \kappa/4$. Thus, to outperform this attack, any algorithm with constant exponential growth will have to grow by at most a factor of 16 for any additional intermediate round. The growth we observe in our experiments is nowhere close to this:

– The running time when going from 1 to 2 internal rounds is increased by a factor of around 46;
– The factor between 2 rounds and the data point where computations for 3 rounds was halted, is more than 80.

This observation, along with the conservative choice of only including two external rounds in these tests, leads us to reasonably conjecture that the complexity of Gröbner basis attacks greatly exceeds the complexity of interpolation attacks.

E.5 Higher-Order Differential Attack

Given a vectorial Boolean function $\mathcal{F}$ over $\mathbb{F}_2^n$ of degree d, the higher-order differential attack [23, 22] exploits the fact that

$$\sum_{x \in \mathfrak{V} + v} x = \sum_{x \in \mathfrak{V} + v} \mathcal{F}(x) = 0$$

for each affine subspace $\mathfrak{V} + v \subseteq \mathbb{F}_2^n$ of dimension strictly bigger than d (i.e., $\dim(\mathfrak{V}) \geq d + 1$). The corresponding attack in the case of a prime field $\mathbb{F}_p$ has been recently proposed in [4]. Since this result is related to the degree of the polynomial that describes the permutation, we claim that the number of rounds necessary to guarantee security against the interpolation attack provides security against this attack as well.

F Preventing Infinitely Long Subspace Trails in Lai–Massey Constructions

As shown in [28], a weakness of the Lai–Massey construction is the possibility to choose a nonzero input difference such that the quadratic function in the $\mathcal{I}$

rounds is not active. Here, we show how to choose the matrix $M_{\mathcal{I}}$ in order to fix this problem.

For reaching this goal, we follow the same strategy proposed for HadesMiMC in [19]. Instead of talking of differences, we deal with subspaces and we make use of the *subspace trail* notation introduced in [17], and recalled in Definition 3. As in the case of HadesMiMC, the choice of the linear layer M plays a crucial role for preventing subspace trails that can cover an arbitrary number of rounds. In this section, we focus on the general case given in Section 5.1, where

$$y_i = x_i + F\left(\sum_{j=0}^{t-1} \lambda_j^{(1)} \cdot x_j, \sum_{j=0}^{t-1} \lambda_j^{(2)} \cdot x_j, \ldots, \sum_{j=0}^{t-1} \lambda_j^{(s)} \cdot x_j\right).$$

We note that this includes both the nonlinear layer in $\mathcal{I}$ of the $\mathcal{P}$ permutation, defined over $\mathbb{F}^4$, and the nonlinear layer in each of the rounds of the $\mathcal{P}_{\mathtt{K}}^{(e)}$ permutation, defined over $\mathbb{F}^8$.

We point out that it is *not* possible that both an arbitrarily long subspace trail exists (e.g., $M^i \times \mathfrak{U} = \mathfrak{U}$ for a certain $i \geq 1$) and the condition $\dim\big(\mathfrak{U} \cap (M^i \times \mathfrak{U})\big) \leq \dim(\mathfrak{U}) - 1$ is satisfied. Before going on, we also recall the concept of infinitely long invariant/iterative subspace trails.

Definition 4 (Infinitely Long Invariant/Iterative Subspace Trail [19]). *Let $(\mathfrak{V}_0, \mathfrak{V}_1, \ldots, \mathfrak{V}_{l-1})$ be a constant-dimensional subspace trail for l rounds. We call this subspace trail an infinitely long iterative subspace trail of period l for the considered scheme if it repeats itself an arbitrary number of times, i.e., if*

$$(\mathfrak{V}_0, \mathfrak{V}_1, \ldots, \mathfrak{V}_{l-1}, \mathfrak{V}_0, \mathfrak{V}_1, \ldots, \mathfrak{V}_{l-1}, \ldots, \mathfrak{V}_0, \mathfrak{V}_1, \ldots, \mathfrak{V}_{l-1}, \ldots)$$

is an infinitely long subspace trail.

F.1 Necessary Condition for Preventing Infinitely Long Subspace Trails

First, we define the subspaces $\mathfrak{X}^{(i)}$ which are the starting points of our analysis.

Definition 5. *For $i \geq 0$ and $t \geq 2$, let $\mathfrak{X}^{(i)} \subseteq \mathbb{F}_p^t$ be the subspace defined as*

$$\begin{aligned}\mathfrak{X}^{(i)} &= \left\{ x \in \mathbb{F}_p^t \;\middle|\; \sum_{l=0}^{t-1} \left(\lambda_l^{(h)} \cdot \left(M^j \times x\right)_l\right) = 0 \in \mathbb{F}^t \text{ for each } j \leq i, 0 \leq h \leq s \right\} \\ &= \bigcap_{h=0}^{s} \left\{ x \in \mathbb{F}_p^t \;\middle|\; \sum_{l=0}^{t-1} \left(\lambda_l^{(h)} \cdot \left(M^j \times x\right)_l\right) = 0 \in \mathbb{F}^t \text{ for each } j \leq i \right\}. \end{aligned} \quad (15)$$

Moreover, let I be the largest $i \geq t-1$ such that $1 \leq \dim(\mathfrak{X}^{(i)}) \leq t-1$ (that is, $\dim(\mathfrak{X}^{(I)}) \geq 1$ and $\dim(\mathfrak{X}^{(I+1)}) = 0$).

By taking a pair of texts in the same coset of $\mathfrak{X}^{(i)}$, the first i rounds are essentially linear, since the input difference of F is always zero. More precisely, the output difference $F(\varDelta_1, \varDelta_2, \ldots, \varDelta_s)$ is zero for the input differences $(\varDelta_1, \varDelta_2, \ldots, \varDelta_s) = (0, 0, \ldots, 0)$. Depending on M, this behavior may repeat for an arbitrary number of rounds, and our goal is to avoid this by choosing M properly.

Inactive *F* Function. We will first focus on the case where F is not active (i.e., the input difference is always zero). Working as in [19], we derive the following result.

Proposition 3. *A subspace $\mathfrak{I} \subset \mathbb{F}_p^t$ defines an infinitely long subspace trail with inactive F if and only if $\mathfrak{I} \subseteq \mathfrak{X}^{(0)}$ and $M \times \mathfrak{I} = \mathfrak{I}$.*

Proof. Clearly, a subspace $\mathfrak{I} \subseteq \mathfrak{X}^{(0)}$ fulfilling $M \times \mathfrak{I} = \mathfrak{I}$ generates an infinitely long invariant subspace trail with inactive F functions, since this function is inactive in the first round and then $\mathfrak{I}$ is repeated infinitely.

Next, we show that, given an infinitely long invariant subspace trail $\mathfrak{I}$ with inactive F functions, it must satisfy $\mathfrak{I} \subseteq \mathfrak{X}^{(0)}$ and $M \times \mathfrak{I} = \mathfrak{I}$. Indeed, $\mathfrak{I} \subseteq \mathfrak{X}^{(0)}$, otherwise F would be active in the first round. Moreover, $\mathfrak{I}$ is invariant if and only if $M \times \mathfrak{I} = \mathfrak{I}$. The result follows immediately.

It remains to show that the nonexistence of infinitely long invariant subspace trails implies the nonexistence of infinitely long iterative subspace trails. For this purpose, we prove that if an infinitely long iterative subspace trail with inactive F functions exists, then an invariant one exists as well. Let $\{\mathfrak{I}, M \cdot \mathfrak{I}, M^2 \cdot \mathfrak{I}, \ldots, M^{l-1} \cdot \mathfrak{I}\}$ be an l-round iterative subspace trail. Let $\mathfrak{I}' := \langle \mathfrak{I}, M \cdot \mathfrak{I}, M^2 \cdot \mathfrak{I}, \ldots, M^{l-1} \cdot \mathfrak{I}\rangle$. By definition, $\mathfrak{I}'$ generates an invariant subspace. Moreover, it is a non-trivial subspace of $\mathbb{F}_p^t$ (that is, $\mathfrak{I}' \subset \mathbb{F}_p^t$) since on such a subspace F is never active by assumption. □

Active *F* Function. In order to provide security, we must also consider infinitely long subspace trails in which the input difference of the F function is nonzero. If such a subspace exists, it would be possible to skip the middle rounds without increasing the degree, which reduces the resistance against algebraic attacks.

To give a concrete example, consider again the case $t = 4$, where $\lambda_0 = \lambda_2 = 1$ and $\lambda_1 = \lambda_3 = -1$. Moreover, let

$$M_{\mathcal{I}} = \begin{pmatrix} \mu_0 & 1 & 1 & 1 \\ 1 & \mu_1 & 1 & 1 \\ 1 & 1 & \mu_1 & 1 \\ 1 & 1 & 1 & \mu_1 \end{pmatrix}$$

for $\mu_0, \mu_1 \in \mathbb{F}_p$. The two-dimensional subspace $\mathfrak{I} = \langle (1, 0, 0, 0), (0, 1, 1, 1) \rangle \subseteq \mathbb{F}_p^4$ generates an infinitely long invariant subspace trail in which F is active. Indeed, given $(x, y, y, y) \in \mathfrak{I}$, note that the input of F is $(x-y, x-y, x-y, x-y)$, which is in general not equal to zero.

In the following, we give a necessary condition that, if satisfied, guarantees that no infinitely long invariant subspace trail with active F functions exists.

Proposition 4. *If a subspace $\mathfrak{I} \subset \mathbb{F}_p^t$ defines an infinitely long invariant subspace trail with active F functions, then*

$$\langle (1, 1, \ldots, 1) \rangle \subseteq \mathfrak{I}.$$

Proof. Let $\mathfrak{I}$ be an invariant subspace trail. For each $x \in \mathfrak{I}$,

$$M \times (x_0 + F(x_0, \ldots, x_{t-1}), x_1 + F(x_0, \ldots, x_{t-1}), \ldots, x_{t-1} + F(x_0, \ldots, x_{t-1}))^T \in \mathfrak{I},$$

or equivalently $M \times x + F(x_0, \ldots, x_{t-1}) \cdot M \times (1, 1, \ldots, 1)^T \in \mathfrak{I}$, i.e.,

$$M \times \mathfrak{I} + \langle M \times (1, 1, \ldots, 1)^T \rangle = \mathfrak{I} \quad \Longrightarrow \quad M \times (\mathfrak{I} + \langle (1, 1, \ldots, 1)^T \rangle) = \mathfrak{I}$$

where M is invertible. Since $\mathfrak{I}$ is an invariant subspace, we have that

$$\dim \left(M \times (\mathfrak{I} + \langle (1, 1, \ldots, 1)^T \rangle) \right) = \dim \left(\mathfrak{I} + \langle (1, 1, \ldots, 1)^T \rangle \right) = \dim(\mathfrak{I}) ,$$

and hence

$$\langle (1, 1, \ldots, 1)^T \rangle \subseteq \mathfrak{I}. \qquad \square$$

Finally, in the case of infinitely long iterative subspace trails with active F functions, we simply reuse the results just given by observing that an infinitely long iterative subspace trail of period l is an infinitely long l-round invariant subspace trail.

F.2 Sufficient Condition for Preventing Infinitely Long Subspace Trails

Note that in the approaches described above, the existence of an infinitely long subspace trail relies on a matrix M for which a nontrivial M-invariant subspace exists. Equivalently, a matrix M for which no M-invariant subspace exists is considered secure. Therefore, we can reuse [19, Proposition 12], i.e., we determine if the minimal polynomial of a given matrix has degree 8 and is irreducible. If this is the case, no infinitely long subspace trails with inactive or active F functions exist, and the same result can also be applied to $M_{\mathcal{E}}$ in $\mathcal{P}^{(\mathrm{B})}$. We refer to Algorithm 2 for the detailed approach. We point out that this approach provides a sufficient condition for preventing infinitely long subspace trails which is not necessary in general.

Algorithm 2: Determining if a given matrix is potentially vulnerable to subspace trails in Hydra.

```
   Data: Matrix $M \in \mathbb{F}_p^{n\times n}$, where $n \in \{4, 8\}$.
   Result: False if there is a nonzero chance of vulnerability, True otherwise.
1  $l \leftarrow n$.
2  $M' \leftarrow M$.
3  for $i \leftarrow 0$ to $l$ do
4      if $\deg(\phi_{M'}) < n$ or $\phi_{M'}$ is not irreducible then
5          return False
6      $M' \leftarrow M \times M'$.
7  return True
```

G Proofs for the Constructions

G.1 Proof of Proposition 1

Proof. As in a Lai–Massey construction, the invertibility follows from the fact that for $i \in \{1, 2, \dots, t\}$

$$\sum_{h=0}^{n-1} \lambda_h^{(i)} y_h = \sum_{h=0}^{n-1} \lambda_h^{(i)} \cdot \left(x_h + F'\left(\sum_{j=0}^{n-1} \lambda_j^{(1)} x_j, \sum_{j=0}^{n-1} \lambda_j^{(2)} x_j, \dots, \sum_{j=0}^{n-1} \lambda_j^{(t)} x_j \right) \right)$$
$$= \sum_{h=0}^{n-1} \lambda_h^{(i)} x_h + \underbrace{\sum_{h=0}^{n-1} \lambda_h^{(i)}}_{=0} \cdot F'\left(\sum_{j=0}^{n-1} \lambda_j^{(1)} x_j, \sum_{j=0}^{n-1} \lambda_j^{(2)} x_j, \dots, \sum_{j=0}^{n-1} \lambda_j^{(t)} x_j \right) = \sum_{h=0}^{n-1} \lambda_h^{(i)} x_h.$$

It follows that for $l \in \{0, 1, \dots, n-1\}$ we have

$$x_l = y_l - F'\left(\sum_{j=0}^{n-1} \lambda_j^{(1)} \cdot y_j, \sum_{j=0}^{n-1} \lambda_j^{(2)} \cdot y_j, \dots, \sum_{j=0}^{n-1} \lambda_j^{(t)} \cdot y_j \right). \qquad \square$$

G.2 Proof of Lemma 1

Proof. The proof follows from the fact that $\mathcal{D}'$ satisfies the requirements of Theorem 1:

(1) $\mathcal{D}'_{d,\alpha}(0) \neq 0$, since $\mathcal{D}'_{d,\alpha}(0) = \frac{d}{d-\lfloor d/2 \rfloor} \cdot \binom{d-\lfloor d/2 \rfloor}{\lfloor d/2 \rfloor} \cdot (-\alpha)^{\lfloor d/2 \rfloor} \neq 0$, and

(2) $\mathcal{D}_{d,\alpha}(z) = z \cdot \mathcal{D}'_{d,\alpha}(z)$ is a permutation due to Theorem 2, since $\gcd(d, p^2-1) = 1$. $\square$

H Details about Security Analysis of $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$

H.1 Details about Statistical Attacks against $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$

Lemma 2. *Let $t \geq 2$. Let $\omega_0, \omega_1, \omega_2, \ldots, \omega_{t-1} \in \mathbb{F}_p \setminus \{0\}$ where $\sum_{h=0}^{t-1} \omega_h = 0$. Let $S^{''} : \mathbb{F}_p^t \to \mathbb{F}_p^t$ be defined as $S^{''}(x) = y$ where*

$$\forall l \in \{0, 1, \ldots, t-1\} : \qquad y_l = x_l + \left(\sum_{h=0}^{t-1} \omega_h \cdot x_h\right)^2 .$$

Then, for each $\Delta_I, \Delta_O \in \mathbb{F}_p^t$:

$$Prob(\Delta_I \to \Delta_O) = \begin{cases} p^{-1} & \text{if } \Delta_O[0] = \Delta_O[1] = \ldots = \Delta_O[t-1] (\neq 0) \\ 0 & \text{otherwise} \end{cases} .$$

Proof. In order to prove such result, we count the number of solutions $x \in \mathbb{F}_p^t$ of $S^{''}(x + \Delta_I) - S^{''}(x) = \Delta_O$, that is,

$$\forall l \in \{0, 1, \ldots, t-1\} : \left(\sum_{h=0}^{t-1} \omega_h \cdot (x_h + \Delta_I[l])\right)^2 - \left(\sum_{h=0}^{t-1} \omega_h \cdot x_h\right)^2 = \Delta_O[l]$$

First of all, such system of equation admits solution **only if** $\Delta_O[l] = \Delta_O[j]$ for each $l, j \in \{0, 1, \ldots, t-1\}$. Secondly, it is not hard to check that each one of the equations in the system is linear in $x_0, x_1, \ldots, x_{t-1}$. Hence, the number of solutions is at most p^{t-1} ($t-1$ variables are free to take any possible value, while the remaining one is fixed), which implies that $\text{Prob}(\Delta_I \to \Delta_O) \leq \frac{p^{t-1}}{p^t} = p^{-1}$. $\square$

We emphasize that the previous result can be easily generalized to the case in which the quadratic function is replaced by a generic degree-d function. In such a case, its maximum differential probability is $(d-1)/p$.

H.2 Linearization Attacks against $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$

We refer to Appendix E.3 for details about the linearization attack.

Density of $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$. Reusing Eq. (12), we can confirm with practical tests that this bound is almost reached in all of the rounds, including the maximum degree being 2^r. This comparison is also shown in Fig. 6.

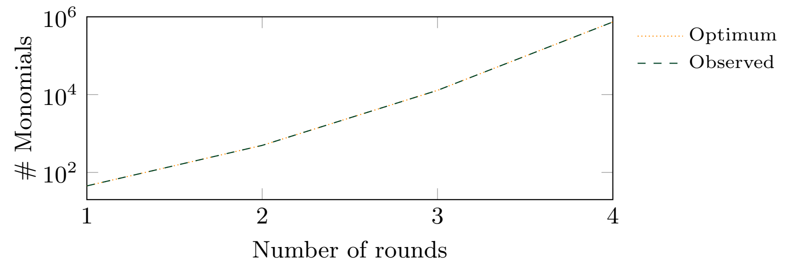


Fig. 6: Comparison of the maximum number of monomials and the observed number of monomials in $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$.

Linearization Attack on $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$. As in Appendix E.3 we will ignore data complexity, in order to arrive at a conservative bound on R_{H} for preventing linearization attacks. In other words, we will assume that the attacker can collect enough inputs/outputs data to directly linearize the system. The degree D is given by $D = 2^{R_{\mathrm{H}}}$ after R_{H} rounds. Since $2^\kappa \le p^2$, it follows that the attack is not feasible if

$$\forall x \in \{0,1\}: \qquad p^x \cdot \left(\binom{12 - x + 2^{R_{\mathrm{H}}}}{12 - x} \right)^2 \ge 2^\kappa \, ,$$

where we consider $\omega = 2$ and where we have 12 variables (namely, the key, y and z). Let $x' = 12 - x$. As before:

$$p^x \cdot \left(\binom{D + x'}{x'} \right)^2 \ge p^x \cdot \left(\frac{\prod_{j=1}^{x'} (D+j)}{x'!} \right)^2 \ge p^x \cdot \left(\frac{(D+1)^{x'}}{x'^{x'}} \right)^2 \ge p^x \cdot \left(\frac{D}{x'} \right)^{2 \cdot x'}$$

where $z^z \ge z!$ for each $z \ge 1$. Hence, this attack is prevented if

$$R_{\mathrm{H}} \ge 3 + \max_{x \in \{0,1\}} \frac{\kappa - x \cdot \log_2(p)}{2 \cdot (12 - x)} + \log_2(12 - x) \, , \qquad (16)$$

where we arbitrary add 3 rounds for destroying possible relations existing between the coefficients of the monomials (due to the fact that the degree-4 function that defines the nonlinear layer is not generic, but has a particular structure).

In the case $\kappa \approx \log_2(p)$, our tests suggest that the number of rounds necessary to satisfy Eq. (10) satisfies Eq. (16) as well.

H.3 Details About Gröbner Basis Attacks Against $\mathcal{P}_{\mathtt{K}}^{(\mathbf{H})}$

This appendix focuses on Gröbner basis attacks where the intermediate rounds of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$ are modeled with extra variables and equations. The final output (after summation-truncation) is assumed to be known. In the following, $D_{\mathrm{reg}}(n_e, n_v)$ denotes the degree of regularity associated with a *quadratic* semi-regular system of n_e equations in n_v variables, as predicted by Eq. (8).

Best Scenario for the Attacker. Each head $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, i)$ included in the modeling increases the number of variables, so the attacker wants to include as few of them as possible. On the other hand, at least two heads are needed to construct an (over-)determined polynomial system. Two choices of pairs stand out: $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, 0)$, $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, 1)$, which has 14 final output entries, and $\mathcal{P}_{K}^{(H)}(\cdot, 0)$, $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, 2)$ which share the matrix $M_{\mathcal{J}}^{0}$ and constants $\lambda_i^{(0)}$. However, the latter modeling only yields 12 final output variables, and the testing we have done suggests that the modeling from $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, 0)$ and $\mathcal{P}_{K}^{(H)}(\cdot, 1)$ is faster to solve. Hence, we will focus on the analysis of this case.

Number of Variables and Equations. Since there is only a single squaring repeated in $S_{\mathcal{J}}^{(i,j)}(\cdot)$, we can, after taking suitable linear combinations, model each intermediate round $\mathcal{J}_j(\cdot, i)$ with one quadratic equation and seven linear equations, in eight new variables. Twelve variables are used for the input y, z, K, and the output of the last round of $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, 0)$ and $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}(\cdot, 1)$ is, when taken together, described by two quadratic equations and six linear equations (as there are only 14 output values after the truncation-summation). We use the linear equations in the system to eliminate variables. Assuming they are all linear independent, we can reduce to a smaller system of $2r$ quadratic equations in $2r - 2$ variables, where r is the number of rounds. This is indeed the case we have observed in all our experiments.

Practical Tests. We have tested the difficulty of computing a grevlex Gröbner basis for these reduced systems, with a varying number of rounds r, over $\mathbb{F}_{7741}$. The tests have been performed running the F_4 algorithm implemented in the computer algebra system MAGMA V2.22-6, on 72 x Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, with 252 GiB RAM. The results are presented in Table 2. 'Step Degrees' lists the degree of the polynomials associated with each step. The maximal step, i.e., the most costly step in terms of running time, is underlined. We present the total running time, as well as the time spent at the maximal step, to give an impression of how the problem scales as r increases. For comparison, we also give the degree of regularity for semi-regular systems of the same size. More precisely, $D_{\mathrm{reg}}(m, n)$ denotes the degree of regularity of a semi-regular system of m quadratic equations in n variables, as described in Section 8.2. For comparison, we also include the running time for random quadratic systems of $2r$ polynomials in $2r - 2$ variables (i.e., the coefficients of all possible monomials up to degree two is a random element of $\mathbb{F}_p$), with a unique solution. Random systems are believed to be semi-regular with high probability (and they indeed turned out to be so in our tests), and the practical hardness of solving them has been studied in cryptographic solving challenges[17]. Note that the unique solution can be directly read from the grevlex Gröbner basis in all our experiments. We can see from Table 2 that the polynomial systems do in general not behave as

[17] https://www.mqchallenge.org/

Table 2: Running time (in seconds) and step degrees in the F_4 algorithm when solving polynomial systems from Hydra over $\mathbb{F}_{7741}$.

r	Step Degrees	Total Time	Time Max Step	$D_{\text{reg}}(2r, 2r-2)$	Time Random
6	2,3,4,5,5,$\underline{5}$	0.18	0.07	6	0.18
7	2,3,4,5,6,$\underline{6}$,5	2.88	1.00	6	2.68
8	2,3,4,5,6,6,7,$\underline{7}$,5	63.44	29.63	7	51.36
9	2,3,4,5,6,7,7,7,$\underline{8}$,7,6	1918.31	1023.13	8	1965.89
10	2,3,4,5,6,7,8,8,8,8,$\underline{9}$,7	82330.73	62141.87	9	78647.85

semi-regular quadratic systems of the same size. Indeed, for $r = 6$, the degree at the maximal step is one less than the associated generic degree of regularity. For $r = 8, 9$ and 10, the degree at the maximal step is equal to the associated D_{reg}, but we observe that the systems emit degree fall polynomials at degree $D_{\text{reg}} - 1$ (which, by definition, would not occur in semi-regular systems). We note that the number of degree fall polynomials found prior to the maximal degree in these cases are relatively small. For $r = 8$, the first step at degree 6 finds 5:1, 6:633 new polynomials, i.e., a single polynomial of degree 5, and 633 polynomials of degree 6. When $r = 9$ the numbers for the first and second steps at degree 7 are 6:16, 7:2432; and 6:5, 7:244. Finally, when $r = 10$ the three first step at degree 8 finds 7:151, 8:8590; 7:92, 8:1946; and 7:9, 8:1476 new polynomials. The degree fall polynomials do not seem to have a major impact on the running time, as in all cases these times are comparable to that of solving random systems of the same size.

The degree fall polynomials in the tests may be explained by an observation that the homogeneous quadratic part of the polynomial systems (after the pre-computed linear reduction) does not contain all variables. Consider the $r = 8$ case, where 8 polynomials had 13 variables in its homogeneous quadratic part. 3 polynomials included 7 variables in their quadratic terms, and the remaining 5 polynomials had 8,9,10,11 and 12 variables respectively. The quadratic part of these polynomials are dense in this number of variables, e.g., the polynomials with 13 variables contain $\binom{13+1}{2}$ terms of degree two. Note that this only holds for the homogeneous quadratic part, as the polynomials contain more variables in their linear forms. In particular, one of the 14 variables only appears in the linear forms.

Adding 3 Extra Rounds. With all these considerations in mind, we find that the complexity of solving a semi-regular system with $2r$ quadratic polynomials in $2r-2$ variables is indeed useful as a baseline, when estimating the complexity of solving $\mathcal{P}_{\mathtt{K}}^{(\mathrm{H})}$. However, we note that the fact that not all polynomials contain every variable, adds a certain structure that an attacker might be able to use when reducing the matrices encountered in F_4. In Table 2 we also observe cases where the algorithm either terminates at degree $D_{\text{reg}} - 1$, or at least finds a small number of degree fall polynomials. To account for these properties, we add three extra rounds on top of this baseline (in addition to the conservative choice of using $\omega = 2$ for the linear algebra constant).

H.4 Computation of the Degree of Regularity given in Eq. (10)

In this Appendix we want to simplify the series in Eq. (8) when applied to the case discussed in the previous subsection, i.e., $n_e - n_v = 2$, $n_e = 2 \cdot R^*$, and $d_i = 2$. Moreover, we want to derive at an easy expression for the first non-positive coefficient, which corresponds to the degree of regularity for the system. These restrictions allow us to rewrite the Hilbert series as

$$H(z) = \frac{\prod_{i=1}^{n_e}(1-z^2)}{(1-z)^{n_v}} = (1-z)^{n_e-n_v} \cdot (1+z)^{n_e} = (1-z)^2 \cdot (1+z)^{n_e},$$

where

$$(1-z)^2 = 1 - 2z + z^2, \qquad \text{and} \qquad (1+z)^{n_e} = \sum_{i=0}^{n_e} \binom{n_e}{i} \cdot z^i \,.$$

Hence,

$$\begin{aligned} H(z) =& \sum_{i=0}^{n_e} \binom{n_e}{i} \cdot z^i - 2 \sum_{i=0}^{n_e} \binom{n_e}{i} \cdot z^{i+1} + \sum_{i=0}^{n_e} \binom{n_e}{i} \cdot z^{i+2} \\ =& 1 + (n_e - 2) \cdot z + \sum_{i=0}^{n_e-2} \left(\binom{n_e}{i+2} - 2 \cdot \binom{n_e}{i+1} + \binom{n_e}{i} \right) \cdot z^{i+2} + \\ & + (n_e - 2) \cdot z^{n_e+1} + z^{n_e+2} \,. \end{aligned} \tag{17}$$

Since

$$\begin{aligned} &\binom{n_e}{i+1} = \binom{n_e+1}{i+1} - \binom{n_e}{i}, \qquad \text{and} \\ &\binom{n_e}{i+2} = \binom{n_e+1}{i+2} - \binom{n_e}{i+1} = \binom{n_e+2}{i+2} - 2 \cdot \binom{n_e+1}{i+1} + \binom{n_e}{i}, \end{aligned}$$

the coefficients in the sum in the right hand side of Eq. (17) can be written out as

$$\begin{aligned} &\binom{n_e}{i+2} - 2 \cdot \binom{n_e}{i+1} + \binom{n_e}{i} = \binom{n_e+2}{i+2} - 4 \cdot \binom{n_e+1}{i+1} + 4 \cdot \binom{n_e}{i} \\ =& \binom{n_e}{i} \cdot \left(\frac{(n_e+2)\cdot(n_e+1)}{(i+2)\cdot(i+1)} - 4\frac{n_e+1}{i+1} + 4 \right) . \end{aligned}$$

Since our goal is to find the first non-positive coefficient of $H(z)$, we are interested in solving

$$\begin{aligned} & 4 \cdot (i+2) \cdot (i+1) - 4 \cdot (n_e+1) \cdot (i+2) + (n_e+2) \cdot (n_e+1) \\ =& 4 \cdot i^2 + i \cdot (8 - 4n_e) + (n_e^2 - 5n_e + 2) = 0 \,, \end{aligned}$$

w.r.t i. That is,

$$i = \frac{4n_e - 8 \pm \sqrt{(8-4n_e)^2 - 16 \cdot (n_e^2 - 5n_e + 2)}}{8} = \frac{n_e - 2 \pm \sqrt{n_e + 2}}{2} .$$

Recalling that $n_e = 2R^*$, and adding 2 to account for the shift z^{i+2} in the sum in the right hand side of Eq. (17), we arrive at

$$D_{\text{reg}} = R^* + 1 - \left\lfloor \frac{\sqrt{2R^* + 2}}{2} \right\rfloor .$$

I Full MPC Benchmarks

Table 3 and Table 4, we give the full MPC benchmarks. Compared to Table 1, we additionally compare Hydra to GMiMC$_{\text{erf}}$ and MiMC-CTR and give benchmarks for more state sizes t. Looking at MiMC, one can observe, that it has a fast online phase performance, but it requires significantly more data transmission between the parties. Furthermore, its large number of multiplications also leads to an expensive offline phase. GMiMC, on the other hand, has a decent performance for very small t, but its number of rounds does not scale well with larger state sizes. Consequently, our benchmark show that it is somewhat competitive for $t = 8$, but its runtime and data transmission explodes with larger t. In any case, Hydra has the faster offline phase performance and also a comparable online phase performance with less data communication compared to MiMC and GMiMC, with the advantage of Hydra growing with the state size t.

J Effect of the linear layer

In Section 9, we discussed, that HadesMiMC has a slow online phase performance when evaluated in a LAN setting due to having many MDS matrix multiplications with bad plain performance. To highlight this effect, we compare Hydra and HadesMiMC (with and without its linear key schedule) to a (insecure) version of HadesMiMC (dubbed HadesMiMC-circ) for which we replaced the random MDS matrices with the matrix $M = \text{circ}(2, 1, 1, \ldots, 1)$. The corresponding matrix-vector multiplications can purely be implemented using additions. In Table 5 one can observe, that this change, as expected, has no effect on the offline phase performance and the communication between the parties. However, the runtime of the online phase became significantly faster and is now twice as fast as the online phase of Hydra. However, combining both phases, one can observe, that Hydra still leads to faster runtimes and less communication due to overall having less multiplications, with larger state sizes further increasing the advantage of Hydra.

Table 3: Online and offline phase performance for evaluating different ciphers with different state sizes t in MPC using a secret shared key. *Prec* is the number of precomputed elements, i.e., multiplication triples, squares, and inverses; *Depth* describes the number of online communication rounds. Runtime averaged over 200 runs. **Bold** values are best values with key schedules, *Italic* the best without.

			Offline			Online		Combined	
Cipher	Rounds	Prec.	Time	Data	Depth	Time	Data	Time	Data
			ms	MB		ms	kB	ms	MB
$t = 8$:									
Hydra	8, 38, 38	**216**	**41.09**	**4.87**	139	7.99	7.19	**49.09**	**4.88**
Ciminion (No KS)[a]	90, 14	*148*	*27.30*	*3.34*	107	*4.10*	*5.02*	*31.39*	*3.35*
Ciminion	90, 14	867	181.50	19.55	735	20.47	28.02	201.97	19.58
HadesMiMC	6, 71	238	45.77	5.37	79	13.84	**5.99**	59.62	5.38
Rescue (No KS)[a]	10	480	97.08	10.83	*33*	5.95	11.80	103.03	10.84
Rescue	10	960	205.17	21.65	***33***	10.44	23.32	215.61	21.68
GMiMC	177	354	68.69	7.99	179	5.80	8.78	74.49	8.00
MiMC	81	1296	270.79	29.23	83	**4.85**	31.38	275.64	29.26
$t = 16$:									
Hydra	8, 38, 38	**254**	**47.74**	**5.73**	139	9.67	8.54	**57.41**	**5.74**
Ciminion (No KS)[a]	90, 14	*208*	*37.32*	*4.69*	111	*4.80*	*7.06*	*42.12*	*4.70*
Ciminion	90, 14	1647	346.28	37.13	1455	37.72	53.11	384.00	37.19
HadesMiMC	6, 71	334	66.60	7.54	79	44.78	**8.42**	111.38	7.55
Rescue (No KS)[a]	10	960	203.46	21.65	*33*	12.00	23.45	215.47	21.68
Rescue	10	1920	412.22	43.30	***33***	26.42	46.49	438.64	43.35
GMiMC	546	1092	228.89	24.63	548	14.55	26.62	243.44	24.66
MiMC	81	2592	582.32	58.46	83	**5.99**	62.62	588.31	58.52
$t = 32$:									
Hydra	8, 38, 38	**330**	**63.69**	**7.44**	139	12.80	**11.22**	**76.49**	**7.46**
Ciminion (No KS)[a]	90, 14	*328*	*63.31*	*7.40*	119	*5.38*	*11.16*	*68.69*	*7.41*
Ciminion	90, 14	3207	808.91	72.30	2895	71.92	103.29	880.83	72.41
HadesMiMC	6, 71	526	108.22	11.87	79	162.15	13.29	270.36	11.88
Rescue (No KS)[a]	10	1920	409.76	43.30	*33*	30.94	46.74	440.70	43.35
Rescue	10	3840	983.05	86.60	***33***	77.30	92.82	1060.35	86.70
GMiMC	2114	4228	1114.55	95.35	2116	49.61	102.14	1164.16	95.46
MiMC	81	5184	1360.13	116.91	83	**9.11**	125.08	1369.24	117.04

[a] Assumes round keys are present, i.e., no key schedule computation in MPC.

Table 4: Online and offline phase performance for evaluating different ciphers with different state sizes t in MPC using a secret shared key. *Prec* is the number of precomputed elements, i.e., multiplication triples, squares, and inverses; *Depth* describes the number of online communication rounds. Runtime averaged over 200 runs. **Bold** values are best values with key schedules, *Italic* the best without.

Cipher	Rounds	Offline			Online			Combined	
		Prec.	Time ms	Data MB	Depth	Time ms	Data kB	Time ms	Data MB
$t = 64$:									
Hydra	8, 38, 38	***520***	***105.23***	***11.73***	139	20.20	***17.82***	**125.43**	***11.75***
Ciminion (No KS)[a]	90, 14	568	117.38	12.81	135	***6.77***	19.35	*124.15*	12.83
Ciminion	90, 14	6327	1819.67	142.64	5775	141.30	203.64	1960.97	142.84
HadesMiMC	6, 71	910	191.50	20.53	79	613.61	23.02	805.11	20.55
Rescue (No KS)[a]	10	3840	989.10	86.60	*33*	97.35	93.34	1086.44	86.70
Rescue	10	7680	2255.42	173.20	***33***	270.98	185.50	2526.40	173.39
GMiMC	8322	16644	5179.08	375.36	8324	196.46	400.63	5375.54	375.76
MiMC	81	10368	3239.21	233.82	83	14.61	250.01	3253.82	234.07
$t = 96$:									
Hydra	8, 38, 38	***672***	***140.01***	***15.15***	139	24.69	***23.19***	***164.70***	***15.18***
Ciminion (No KS)[a]	90, 14	808	165.58	18.22	151	***8.55***	27.54	174.13	18.25
Ciminion	90, 14	9447	2809.31	212.98	8655	208.44	303.99	3017.75	213.28
HadesMiMC	6, 71	1294	275.63	29.19	79	1402.92	32.74	1678.56	29.22
Rescue (No KS)[a]	10	5760	1602.94	129.90	*33*	205.80	139.93	1808.74	130.04
Rescue	10	11520	3545.07	259.80	***33***	584.85	278.17	4129.92	260.08
GMiMC	18626	37252	11607.40	840.11	18628	457.22	895.74	12064.62	841.01
MiMC	81	15552	4801.74	350.73	83	21.74	374.94	4823.48	351.11
$t = 128$:									
Hydra	8, 38, 38	***862***	***178.32***	***19.44***	139	32.19	***29.78***	***210.51***	***19.47***
Ciminion (No KS)[a]	90, 14	1048	220.69	23.63	167	***9.96***	35.74	230.65	23.67
Ciminion	90, 14	12567	3842.51	283.32	11535	272.65	404.34	4115.16	283.72
HadesMiMC	6, 71	1678	361.24	37.85	79	2443.52	42.47	2804.76	37.89
Rescue (No KS)[a]	10	7680	2284.69	173.20	*33*	359.10	186.52	2643.79	173.39
Rescue	10	15360	4740.96	346.40	***33***	1022.45	370.84	5763.41	346.77
GMiMC	33026	66052	20802.30	1489.61	33028	849.75	1587.45	21652.05	1491.20
MiMC	81	20736	6442.85	467.64	83	27.24	499.86	6470.09	468.14

[a] Assumes round keys are present, i.e., no key schedule computation in MPC.

Table 5: Online and offline phase performance for evaluating different ciphers with different state sizes t in MPC using a secret shared key. *Prec* is the number of precomputed elements, i.e., multiplication triples, squares, and inverses; *Depth* describes the number of online communication rounds. Runtime averaged over 200 runs. **Bold** values are best values with key schedules, *Italic* the best without.

Cipher	Rounds	Prec.	Offline Time	Offline Data	Depth	Online Time	Online Data	Combined Time	Combined Data
			ms	MB		ms	kB	ms	MB
$t = 8$:									
Hydra	8, 38, 38	***216***	***41.09***	***4.87***	139	7.99	7.19	49.09	***4.88***
HadesMiMC (No KS)[a]	6, 71	238	45.54	5.37	*79*	8.81	*5.99*	54.35	5.38
HadesMiMC	6, 71	238	45.77	5.37	***79***	13.84	***5.99***	59.62	5.38
HadesMiMC-circ (No KS)[a]	6, 71	238	44.30	5.37	*79*	*3.79*	*5.99*	*48.10*	5.38
HadesMiMC-circ	6, 71	238	44.59	5.37	***79***	**3.88**	***5.99***	**48.47**	5.38
$t = 16$:									
Hydra	8, 38, 38	***254***	***47.74***	***5.73***	139	9.67	8.54	***57.41***	***5.74***
HadesMiMC (No KS)[a]	6, 71	334	64.85	7.54	*79*	24.72	*8.42*	89.57	7.55
HadesMiMC	6, 71	334	66.60	7.54	***79***	44.78	***8.42***	111.38	7.55
HadesMiMC-circ (No KS)[a]	6, 71	334	66.70	7.54	*79*	*4.65*	*8.42*	71.35	7.55
HadesMiMC-circ	6, 71	334	64.69	7.54	***79***	**4.97**	***8.42***	69.66	7.55
$t = 32$:									
Hydra	8, 38, 38	***330***	***63.69***	***7.44***	139	12.80	***11.22***	***76.49***	***7.46***
HadesMiMC (No KS)[a]	6, 71	526	108.21	11.87	*79*	84.74	13.29	192.95	11.88
HadesMiMC	6, 71	526	108.22	11.87	***79***	162.15	13.29	270.36	11.88
HadesMiMC-circ (No KS)[a]	6, 71	526	108.77	11.87	*79*	*6.02*	13.29	114.86	11.88
HadesMiMC-circ	6, 71	526	109.42	11.87	***79***	**6.09**	13.29	115.44	11.88
$t = 64$:									
Hydra	8, 38, 38	***520***	***105.23***	***11.73***	139	20.20	***17.82***	***125.43***	***11.75***
HadesMiMC (No KS)[a]	6, 71	910	187.95	20.53	*79*	316.71	23.02	504.66	20.55
HadesMiMC	6, 71	910	191.50	20.53	***79***	613.61	23.02	805.11	20.55
HadesMiMC-circ (No KS)[a]	6, 71	910	191.96	20.53	*79*	*9.57*	23.02	201.53	20.55
HadesMiMC-circ	6, 71	910	191.63	20.53	***79***	**9.90**	23.02	201.52	20.55
$t = 128$:									
Hydra	8, 38, 38	***862***	***178.32***	***19.44***	139	32.19	***29.78***	***210.51***	***19.47***
HadesMiMC (No KS)[a]	6, 71	1678	362.36	37.85	*79*	1222.09	42.47	1584.45	37.89
HadesMiMC	6, 71	1678	361.24	37.85	***79***	2443.52	42.47	2804.76	37.89
HadesMiMC-circ (No KS)[a]	6, 71	1678	370.05	37.85	*79*	*16.33*	42.47	386.70	37.89
HadesMiMC-circ	6, 71	1678	358.10	37.85	***79***	**16.66**	42.47	374.43	37.89

[a] Assumes round keys are present, i.e., no key schedule computation in MPC.

References for Supplementary Material

[1] T. Baignères, J. Stern, and S. Vaudenay. "Linear Cryptanalysis of Non Binary Ciphers". In: *SAC*. Vol. 4876. LNCS. 2007, pp. 184–211.

[2] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. V. Assche, and R. V. Keer. "Farfalle: parallel permutation-based cryptography". In: *IACR Trans. Symmetric Cryptol.* 2017.4 (2017), pp. 1–38.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. *The KECCAK reference.* https://keccak.team/files/Keccak-reference-3.0.pdf. 2011.

[4] T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. "Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems". In: *CRYPTO*. Vol. 12172. LNCS. 2020, pp. 299–328.

[5] E. Biham, A. Biryukov, and A. Shamir. "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials". In: *EUROCRYPT*. Vol. 1592. LNCS. 1999, pp. 12–23.

[6] E. Biham and A. Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *CRYPTO*. Vol. 537. LNCS. 1990, pp. 2–21.

[7] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer, 1993.

[8] A. Bogdanov and V. Rijmen. "Linear hulls with correlation zero and linear cryptanalysis of block ciphers". In: *Des. Codes Cryptogr.* 70.3 (2014), pp. 369–383.

[9] N. Bordes, J. Daemen, D. Kuijsters, and G. V. Assche. "Thinking Outside the Superbox". In: *CRYPTO*. Vol. 12827. LNCS. 2021, pp. 337–367.

[10] C. Cid, L. Grassi, A. Gunsing, R. Lüftenegger, C. Rechberger, and M. Schofnegger. *Influence of the Linear Layer on the Growth of the Algebraic Degree in SP-Networks.* Cryptology ePrint Archive, Report 2020/536. https://ia.cr/2020/536. 2020.

[11] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. "Efficient algorithms for solving overdefined systems of multivariate polynomial equations". In: *International Conference on the Theory and Applications of Cryptographic Techniques.* Springer. 2000, pp. 392–407.

[12] J. Daemen, L. R. Knudsen, and V. Rijmen. "The Block Cipher Square". In: *FSE*. Vol. 1267. LNCS. 1997, pp. 149–165.

[13] J. Daemen and V. Rijmen. "The Wide Trail Design Strategy". In: *Cryptography and Coding - IMA International Conference 2001.* Vol. 2260. LNCS. 2001, pp. 222–238.

[14] J. Ding, A. Petzoldt, and D. S. Schmidt. *Multivariate Public Key Cryptosystems.* https://doi.org/10.1007/978-1-0716-0987-3. Springer, 2020.

[15] L. Grassi. "Mixture Differential Cryptanalysis: a New Approach to Distinguishers and Attacks on round-reduced AES". In: *IACR Trans. Symmetric Cryptol.* 2018.2 (2018), pp. 133–160.

[16] L. Grassi, Y. Hao, C. Rechberger, M. Schofnegger, R. Walch, and Q. Wang. *A New Feistel Approach Meets Fluid-SPN: Griffin for Zero-Knowledge Applications.* Cryptology ePrint Archive, Report 2022/??? https://ia.cr/2022/???. 2022.

[17] L. Grassi, C. Rechberger, and S. Rønjom. "Subspace Trail Cryptanalysis and its Applications to AES". In: *IACR Trans. Symmetric Cryptol.* 2016.2 (2016), pp. 192–225.

[18] L. Grassi, C. Rechberger, and S. Rønjom. “A New Structural-Differential Property of 5-Round AES”. In: *EUROCRYPT*. Vol. 10211. LNCS. 2017, pp. 289–317.

[19] L. Grassi, C. Rechberger, and M. Schofnegger. “Proving Resistance Against Infinitely Long Subspace Trails: How to Choose the Linear Layer”. In: *IACR Trans. Symmetric Cryptol.* 2021.2 (2021), pp. 314–352.

[20] T. Jakobsen and L. R. Knudsen. “The Interpolation Attack on Block Ciphers”. In: *FSE*. Vol. 1267. LNCS. 1997, pp. 28–40.

[21] N. Keller and A. Rosemarin. “Mind the Middle Layer: The HADES Design Strategy Revisited”. In: *EUROCRYPT*. Vol. 12697. LNCS. 2021, pp. 35–63.

[22] L. R. Knudsen. “Truncated and Higher Order Differentials”. In: *FSE*. Vol. 1008. LNCS. 1994, pp. 196–211.

[23] X. Lai. “Higher Order Derivatives and Differential Cryptanalysis”. In: *Communications and Cryptography: Two Sides of One Tapestry*. Springer US, 1994, pp. 227–233.

[24] G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. “A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack”. In: *CRYPTO*. Vol. 6841. LNCS. 2011, pp. 206–221.

[25] G. Leander, B. Minaud, and S. Rønjom. “A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro”. In: *EUROCRYPT*. Vol. 9056. LNCS. 2015, pp. 254–283.

[26] G. Leander, C. Tezcan, and F. Wiemer. “Searching for Subspace Trails and Truncated Differentials”. In: *IACR Trans. Symmetric Cryptol.* 2018.1 (2018), pp. 74–100.

[27] M. Matsui. “Linear Cryptanalysis Method for DES Cipher”. In: *EUROCRYPT*. Ed. by T. Helleseth. Vol. 765. LNCS. 1993, pp. 386–397.

[28] S. Vaudenay. “On the Lai-Massey Scheme”. In: *ASIACRYPT*. Vol. 1716. LNCS. 1999, pp. 8–19.

[29] D. A. Wagner. “The Boomerang Attack”. In: *FSE*. Vol. 1636. LNCS. 1999, pp. 156–170.