Corrigendum to “Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over Fnp : Application to Poseidon”

IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2026, No. 1, pp. 527–532. DOI:10.46586/tosc.v2026.i1.527-532

Corrigendum to "Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over $\mathbb{F}_p^n$: Application to Poseidon"

Matilda Urani[1] and Lorenzo Grassi[2]

[1] Politecnico di Torino, Torino, Italy
matilda.urani@polito.it
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
l.grassi@tue.nl

Abstract. Neptune is a hash function proposed by Grassi et al. at ToSC 2022(3) for Zero-Knowledge (ZK) applications. In this note, we show that the linear layer of Neptune's external rounds fails to guarantee the maximum growth of the degree, potentially affecting the security of Neptune against algebraic attacks. Here, we formally address this problem, by identifying sufficient conditions that ensure the expected degree growth is maintained.

Keywords: Neptune, Linear Layer, Maximum Degree

1 Introduction and Motivation

Neptune is a sponge hash function proposed by Grassi et al. [GOPS22] at ToSC 2022(3) for Zero-Knowledge (ZK) applications. The design of Neptune is based on the Hades design strategy [GLR+20], a generalization of SPN-type schemes, which combines both rounds with full S-Box layer and rounds with partial S-Box layer to achieve security and good performance. In particular, Neptune is a variant of the hash function Poseidon/Poseidon2 [GKR+21, GKS23], with the main difference relying on the external full rounds, as recalled later on.

The design of ZK-friendly schemes requires particular care to prevent algebraic attacks, which exploit the low degree and/or low density of the system of equations that describes the primitive to break it. A common countermeasure is to make the polynomial expressing the function of high enough (potentially, maximum) degree.

In this context, the choice of matrix defining Neptune's linear layer becomes crucial. We show that the conditions in [GOPS22] for the linear layer of the external full rounds can provide room for flaws of this kind, as they do not necessarily ensure that the function achieves its maximum possible degree. Note that if the polynomial representation of the scheme exhibits a lower algebraic degree than expected, the corresponding security analysis may no longer be considered reliable, and the likelihood that an adversary can mount a successful attack may be higher. To address this, we formally identify sufficient conditions that ensure the expected degree growth is maintained. In App. A, we also provide a Sage script that takes in input a candidate for the linear layer of Neptune, and checks if these conditions are satisfied or not. Finally, we show that the probability of finding a matrix that satisfies these criteria is sufficiently high, thereby ensuring that the security analysis in [GOPS22] remains reliable in practice.

Received: 2025-11-23 Accepted: 2026-01-23 Published: 2026-03-16

1.1 Neptune's External Rounds

For the goal of this paper, we limit ourselves to recall the details of the external round of Neptune. The round function consists of two main components: a nonlinear S-Box layer and a subsequent linear mixing layer, followed by a random constant addition. Here, we introduce the notation $x^{(r)}$, $y^{(r)}$, and $z^{(r)}$ to denote, respectively, the input, the output of the S-Box, and the output of the linear layer at round r.

Initial State. Let $x^{(r)} = (x_0^{(r)}, x_1^{(r)}, \ldots, x_{t-1}^{(r)}) \in \mathbb{F}_p^t$ denote the state at the input of round r of dimension $t = 2 \cdot t' > 2$, where $\mathbb{F}_p$ is the underlying field, typically with a prime $p > 2^{63}$, and t' denotes the number of pairs of field elements in the state.

S-Box. Let $\alpha, \gamma \in \mathbb{F}_p \setminus \{0\}$ be fixed constants. The S-Box function of Neptune is defined over $\mathbb{F}_p^2$ as a combination of (1st) the Lai-Massey scheme of degree 2 defined as $(w_0, w_1) \mapsto (\alpha \cdot w_0 + (w_0 - w_1)^2, \alpha \cdot w_1 + (w_0 - w_1)^2)$, (2nd) a matrix multiplication with $[2, 1; 1, 3]$ and a round constant addition $[\gamma, 0]$, and (3rd) another application of the Lai-Massey scheme just defined.

With these ingredients, the output of the S-Box $S^{(E)} : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is given by

$$y^{(r)} = S^{(E)}(x^{(r)}) = S'(x_0^{(r)}, x_1^{(r)}) \| S'(x_2^{(r)}, x_3^{(r)}) \| \ldots \| S'(x_{t-2}^{(r)}, x_{t-1}^{(r)}) \,,$$

where $S'(x_{2i}^{(r)}, x_{2i+1}^{(r)}) = y_{2i}^{(r)} \| y_{2i+1}^{(r)}$ with

$$y_{2i}^{(r)} = \alpha^2(2x_{2i}^{(r)} + x_{2i+1}^{(r)}) + 3\alpha(x_{2i}^{(r)} - x_{2i+1}^{(r)})^2 + (\gamma + \alpha(x_{2i}^{(r)} - 2x_{2i+1}^{(r)}) - (x_{2i}^{(r)} - x_{2i+1}^{(r)})^2)^2,$$
$$y_{2i+1}^{(r)} = \alpha^2(x_{2i}^{(r)} + 3x_{2i+1}^{(r)}) + 4\alpha(x_{2i}^{(r)} - x_{2i+1}^{(r)})^2 + (\gamma + \alpha(x_{2i}^{(r)} - 2x_{2i+1}^{(r)}) - (x_{2i}^{(r)} - x_{2i+1}^{(r)})^2)^2.$$

Linear Layer. The S-Box output $y^{(r)}$ is then processed by a linear transformation specified by two independent MDS matrices $M', M'' \in \mathbb{F}_p^{\frac{t}{2} \times \frac{t}{2}} \equiv \mathbb{F}_p^{t' \times t'}$, applied respectively to the even and odd coordinates. The output $z^{(r)}$ of the linear layer at round r is given by

$$z_{2i}^{(r)} = \sum_{j=0}^{t'-1} M'_{i,j} \cdot y_{2j}^{(r)} \,, \qquad z_{2i+1}^{(r)} = \sum_{j=0}^{t'-1} M''_{i,j} \cdot y_{2j+1}^{(r)} \,.$$

1.2 Degree Tracking

As the degree of each external full round of Neptune is 4, the maximum degree that these polynomials can attain at round r is 4^r. Ideally, one would hope that this maximal degree is always reached because it offers greater resistance to algebraic attacks. However, this is not always the case, as the effective degree is highly dependent on the choice of the matrices M' and M''. In [GOPS22] conditions on the entries of these matrices are proposed. Specifically, the authors suggest using MDS matrices such that:

1. $M' \neq m \cdot M''$ for each $m \in \mathbb{F}_p$, and
2. $M'_{i,j} \neq M''_{i,j}$ for each $i, j \in \{0, \ldots, t' - 1\}$.

(We limit ourselves to point out that Neptune's designers do not propose a formal rigorous argument to support these two conditions.) However, we observed that these conditions are not sufficient to guarantee an exponential growth of the degree with respect to the number of rounds. In fact, as we demonstrate below, there exist matrices satisfying the constraints in [GOPS22] but for which the degree growth is not maximal.

Example. Fix $\alpha = 1$ and $\gamma = 1$. Given the matrices $M', M'' \in \mathbb{F}_p^{2\times 2}$, for $p = 2^{64} - 2^{32} + 1$, defined as

$$M' = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad \text{and} \qquad M'' = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

we experimentally verified that the degree growth does not reach its expected value, observing a maximum degree of only $56 < 4^3 = 64$ after three rounds[1].

2 Our Contribution: New Conditions for the Linear Layer of Neptune's External Rounds

We now present possible requirements on M' and M'' that are sufficient to maximize the degree growth throughout the external Neptune rounds. Note that the proposed constraints are independent of the parameters γ or α, as they do not affect the maximum degree of the round output. Moreover, the conditions we derive do not depend on the specific choice of the matrix $[2, 1; 1, 3]$ [GOPS22]. Indeed, even if a different choice for this matrix may alter the coefficients of the highest-degree monomials in the variables x_i, they do not affect their cancellation.

The following notation will be used for presenting our results. Let $\mu^{(r)}_{2i,j}, \mu^{(r)}_{2i+1,j}$ be defined by

$$\mu^{(r)}_{2i,j} := \sum_{k=0}^{t'-1} \mu^{(1)}_{2i,k}(\mu^{(r-1)}_{2k,j} - \mu^{(r-1)}_{2k+1,j})^4, \qquad \mu^{(r)}_{2i+1,j} := \sum_{k=0}^{t'-1} \mu^{(1)}_{2i+1,k}(\mu^{(r-1)}_{2k,j} - \mu^{(r-1)}_{2k+1,j})^4,$$

for $r \geq 2$, where $\mu^{(1)}_{2i,j} = M'_{i,j}$ and $\mu^{(1)}_{2i+1,j} := M''_{i,j}$.

Proposition 1. *Let $r > 1$ be the number of external rounds. Let $x^{(1)}$ be the initial state and $z^{(r)}$ the output of the r^{th} round, where each coordinate admits a polynomial representation in terms of the coordinates of $x^{(1)}$.*
The degree of each polynomial attains its maximum value, i.e., 4^r, if the following conditions hold for each $i, j = 0, \dots t' - 1$:

$$\mu^{(r)}_{2i,j}, \mu^{(r)}_{2i+1,j} \neq 0,$$
$$\mu^{(r-1)}_{2i,j} - \mu^{(r-1)}_{2i+1,j} \neq 0.$$

Before proving this result, we emphasize that these conditions are satisfied with high probability for large $p > 2^{63}$. Indeed, assuming these conditions to be independent,[2] they are satisfied for r rounds with approximately probability equal to $\left(1 - \frac{1}{p}\right)^{3r\cdot t'^2} \approx 1 - \frac{3r\cdot t^2}{4\cdot p}$, indicating that matrices meeting these security requirements can be easily identified.

Proof. For the purposes of our analysis, we focus exclusively on the highest-degree terms. With a slight abuse of notation, we will therefore use $y^{(r)}$ and $z^{(r)}$ to denote not the full output polynomials, but only the terms of the form $(x^{(1)}_{2i} - x^{(1)}_{2i+1})^{4^r}$, as they govern the maximum resulting degree at round r.

For each round $r > 1$, we require that the coefficients of $(x^{(1)}_{2i} - x^{(1)}_{2i+1})^{4^r}$ do not vanish, from which the conditions of the proposition follow straightforwardly.

[1] These results are obtained by computationally evaluating three rounds of Neptune using SageMath.
[2] We checked by experiments that this assumption is usually fulfilled.

Round 1. Let $x^{(1)}$ be the initial state. Since we are only interested in tracking $(x_{2i}^{(1)} - x_{2i+1}^{(1)})^4$, we can simplify the output to the S-Boxes as

$$y_{2i}^{(1)} = y_{2i+1}^{(1)} \cong \left(x_{2i}^{(1)} - x_{2i+1}^{(1)}\right)^4, \quad i = 0, \ldots, t' - 1\,,$$

where $\cong$ focuses on some monomials crucial for our analysis (remember that we are interested in sufficient conditions only, hence, we neglect all other terms that are irrelevant for our analysis). Given $y^{(1)}$, the output of the linear layer can be expressed as

$$z_{2i}^{(1)} = \sum_{j=0}^{t'-1} M'_{i,j} \cdot y_{2j}^{(1)} \cong \sum_{j=0}^{t'-1} \mu_{2i,j}^{(1)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^4\,,$$
$$z_{2i+1}^{(1)} = \sum_{j=0}^{t'-1} M''_{i,j} \cdot y_{2j+1}^{(1)} \cong \sum_{j=0}^{t'-1} \mu_{2i+1,j}^{(1)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^4\,,$$

where each $\mu_{2i,j}^{(1)}, \mu_{2i+1,j}^{(1)}$ assume values in $\mathbb{F}_p \setminus \{0\}$, by definition of M', M''. Each $z_i^{(1)}$ is a sum of terms $(x_{2j}^{(1)} - x_{2j+1}^{(1)})^4$, which never repeat. Therefore the maximum degree is always 4 and no additional condition on the matrices entries is needed.

Round 2. Let $x^{(2)} = z^{(1)}$ be the initial state of the second round. Following the application of the S-box, the terms of interest for our analysis are

$$y_{2i}^{(2)} = y_{2i+1}^{(2)} \cong (x_{2i}^{(2)} - x_{2i+1}^{(2)})^4 \cong \left(\sum_{j=0}^{t'-1} (\mu_{2i,j}^{(1)} - \mu_{2i+1,j}^{(1)})(x_{2j}^{(1)} - x_{2j+1}^{(1)})^4\right)^4,$$

and since our goal is to ensure that the polynomials attain their maximum degree, we can simplify the analysis by considering only

$$y_{2i}^{(2)} = y_{2i+1}^{(2)} \cong \sum_{j=0}^{t'-1} (\mu_{2i,j}^{(1)} - \mu_{2i+1,j}^{(1)})^4 (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^2}\,.$$

As the output of the linear layer is given by $z_{2i}^{(2)} = \sum_{k=0}^{t'-1} \mu_{2i,k}^{(1)} y_{2k}^{(2)}$ and $z_{2i+1}^{(2)} = \sum_{k=0}^{t'-1} \mu_{2i+1,k}^{(1)} y_{2k+1}^{(2)}$, by substituting $y^{(2)}$ and by rearranging the terms, we obtain

$$z_{2i}^{(2)} \cong \sum_{j=0}^{t'-1} \left(\sum_{k=0}^{t'-1} \mu_{2i,k}^{(1)} (\mu_{2k,j}^{(1)} - \mu_{2k+1,j}^{(1)})^4\right) (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^2} = \sum_{j=0}^{t'-1} \mu_{2i,j}^{(2)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^2},$$
$$z_{2i+1}^{(2)} \cong \sum_{j=0}^{t'-1} \left(\sum_{k=0}^{t'-1} \mu_{2i+1,k}^{(1)} (\mu_{2k,j}^{(1)} - \mu_{2k+1,j}^{(1)})^4\right) (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^2} = \sum_{j=0}^{t'-1} \mu_{2i+1,j}^{(2)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^2}.$$

Round r. By inductive hypothesis we can assume that

$$x_{2i}^{(r)} = z_{2i}^{(r-1)} \cong \sum_{j=0}^{t'-1} \mu_{2i,j}^{(r-1)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^{r-1}},\ x_{2i+1}^{(r)} = z_{2i+1}^{(r-1)} \cong \sum_{j=0}^{t'-1} \mu_{2i+1,j}^{(r-1)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^{r-1}}.$$

Following the application of the S-box, the terms of interest for our analysis are

$$y_{2i}^{(r)} = y_{2i+1}^{(r)} \cong (x_{2i}^{(r)} - x_{2i+1}^{(r)})^4 \cong \left(\sum_{j=0}^{t'-1} (\mu_{2i,j}^{(r-1)} - \mu_{2i+1,j}^{(r-1)})(x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^{r-1}}\right)^4,$$

which, as in round 2, can be simplified by considering

$$y_{2i}^{(r)} = y_{2i+1}^{(r)} \cong \sum_{j=0}^{t'-1} (\mu_{2i,j}^{(r-1)} - \mu_{2i+1,j}^{(r-1)})^4 (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^r}.$$

The output of the linear layer will then be given by $z_{2i}^{(r)} = \sum_{k=0}^{t'-1} \mu_{2i,k}^{(1)} y_{2k}^{(r)}$ and $z_{2i+1}^{(r)} = \sum_{k=0}^{t'-1} \mu_{2i+1,k}^{(1)} y_{2k+1}^{(r)}$, which can be written as

$$z_{2i}^{(r)} \cong \sum_{j=0}^{t'-1} \left(\sum_{k=0}^{t'-1} \mu_{2i,k}^{(1)} (\mu_{2k,j}^{(r-1)} - \mu_{2k+1,j}^{(r-1)})^4 \right) (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^r} = \sum_{j=0}^{t'-1} \mu_{2i,j}^{(r)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^r},$$

$$z_{2i+1}^{(r)} \cong \sum_{j=0}^{t'-1} \left(\sum_{k=0}^{t'-1} \mu_{2i+1,k}^{(1)} (\mu_{2k,j}^{(r-1)} - \mu_{2k+1,j}^{(r-1)})^4 \right) (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^r} = \sum_{j=0}^{t'-1} \mu_{2i+1,j}^{(r)} (x_{2j}^{(1)} - x_{2j+1}^{(1)})^{4^r},$$

thereby concluding the proof. □

Acknowledgments

Authors thank Katharina Koschatko for pointing out the problem for the linear layer of Neptune's external rounds. Lorenzo Grassi was supported by the European Research Council (ERC), grant number 101160608 "SYMPZON". Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security Symposium*, pages 519–535. USENIX Association, 2021.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A Faster Version of the Poseidon Hash Function. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023*, volume 14064 of *LNCS*, pages 177–203. Springer, 2023.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 674–704. Springer, 2020.

[GOPS22] Lorenzo Grassi, Silvia Onofri, Marco Pedicini, and Luca Sozzi. Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over $\mathbb{F}_p^n$ – Application to Poseidon. *IACR Trans. Symmetric Cryptol.*, 2022(3):20–72, 2022.

A Sage Code

```
# Checks matrices Mp, Mpp over r rounds and prime p.
# Returns true if conditions hold for all rounds, false otherwise.

def verify_m_matrices(Mp, Mpp, r, p):
    tp = Mp.nrows()
    Fp = GF(p)
    mup_prec = Mp
    mupp_prec = Mpp

    for s in range(2, r+1):
        mup_succ = Matrix(Fp, tp, tp, 0)
        mupp_succ = Matrix(Fp, tp, tp, 0)
        for i in range(tp):
            for j in range(tp):
                if mup_prec[i,j] == mupp_prec[i,j]:
                    return False
                for k in range(tp):
                    diff4 = (mup_prec[k,j] - mupp_prec[k,j])^4
                    mup_succ[i,j] += Mp[i,k] * diff4
                    mupp_succ[i,j] += Mpp[i,k] * diff4
                if mup_succ[i,j] == 0 or mupp_succ[i,j] == 0:
                    return False
        mup_prec, mupp_prec = mup_succ, mupp_succ
    return True



# Example usage

# Parameters
p = 2^64 - 2^32 + 1
t = 4
tprime = t // 2
r = 5
Fp = GF(p)

Mp = Matrix(Fp, tprime, tprime, [1, -45, 2, -2])
Mpp = Matrix(Fp, tprime, tprime, [3, -36, 1, -1])
print(verify_m_matrices(Mp, Mpp, r, p))
```