Rasta: A cipher with low ANDdepth and few ANDs per bit

Christoph Dobraunig[1], Maria Eichlseder[1], Lorenzo Grassi[1], Virginie Lallemand[2], Gregor Leander[2], Eik List[3], Florian Mendel[4], and Christian Rechberger[1]

[1] Graz University of Technology, Austria
firstname.lastname@iaik.tugraz.at
[2] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
firstname.lastname@rub.de
[3] Bauhaus-Universität Weimar, Germany
eik.list@uni-weimar.de
[4] Infineon Technologies AG, Germany
florian.mendel@infineon.com

Abstract. Recent developments in multi party computation (MPC) and fully homomorphic encryption (FHE) promoted the design and analysis of symmetric cryptographic schemes that minimize multiplications in one way or another. In this paper, we propose with Rasta a design strategy for symmetric encryption that has ANDdepth d and at the same time only needs d ANDs per encrypted bit. Even for very low values of d between 2 and 6 we can give strong evidence that attacks may not exist. This contributes to a better understanding of the limits of what concrete symmetric-key constructions can theoretically achieve with respect to AND-related metrics, and is to the best of our knowledge the first attempt that minimizes both metrics simultaneously. Furthermore, we can give evidence that for choices of d between 4 and 6 the resulting implementation properties may well be competitive by testing our construction in the use-case of removing the large ciphertext-expansion when using the BGV scheme.

Keywords: Symmetric encryption, ASASA, homomorphic encryption, multiplicative complexity, multiplicative depth

1 Introduction

In this paper we study symmetric encryption primitives with few AND gates. This firstly feeds on the curiosity about how few AND gates a cryptographic primitive can have for which we do not know attacks or are otherwise able to argue about its security. But secondly, this is motivated by various new developments in applied and theoretical cryptography where AND-related properties are of great interest: Foundational theoretical results on the complexity of PRGs, PRFs, and cryptographic hashes include [6,5]. On the more practical side, constructions with few AND gates were shown to positively affect the cost of countermeasures against side-channel attacks [39], throughput and latency of various applications of secure multiparty-computation protocols [4,38], verification time of SNARKs [2], the cost to avoid ciphertext-expansion in homomorphic encryption schemes [4,20,49], or reducing the signature size of signature schemes based on Sigma-protocols [16,21,27,44,53].

In general, we may be interested in three different metrics. One metric refers to what is commonly called multiplicative complexity (MC), which is simply the number of multiplications (in our case AND gates) in a circuit, see e.g. [17]. A natural variant in the context of encryption schemes is the number of AND gates per encrypted bit (MC/bit). The third metric refers to the multiplicative depth of the circuit, which we will subsequently call ANDdepth.

1.1 Motivating applications

There are many examples where only the number of multiplications matters (perhaps together with the size of the ring in which they operate). SNARKs, protocols for secure multiparty communication based on Yao's garbled circuits, or Sigma-protocol signature schemes come to mind. However, there are also a number of applications where both the ANDdepth and the number of multiplications matter simultaneously, such as the following two important examples.

Preventing ciphertext expansion in homomorphic encryption schemes. All known fully/somewhat homomorphic encryption schemes come with significant, often prohibitive ciphertext expansion. To prevent the thousand-fold to million-fold ciphertext expansion in (F)HE schemes, a decryption circuit of a symmetric encryption scheme has to be homomorphically evaluated in addition to the actual computations on the ciphertext. The downside of this approach is that application-specific operations on the ciphertext become more costly, as the decryption circuit of the cipher always needs to be evaluated as well.

To prevent bootstrapping, we need to choose the FHE parameters generously enough to accommodate all additional noise from the decryption circuit. This is linked to the homomorphic capacity of a concrete instantiation of an FHE scheme, i.e., the number of operations on the ciphertext before an expensive bootstrapping operation is needed. All known candidates for FHE schemes are using noise-based cryptography. Each operation on the homomorphically encrypted ciphertext incurs an increase in the noise. In many schemes, the noise level grows fast with the multiplicative depth of the circuit [19,22]. Hence, symmetric encryption scheme proposals aiming for these types of applications minimize first of all the ANDdepth.

While the cost of the application-specific homomorphic operations only depends on the ANDdepth of the cipher, the cost of evaluating the additional decryption circuit itself primarily depends on the number of multiplications. Thus, the number of AND computations is also a relevant metric.

Applications of secure multiparty computation protocols. There are various classes of practically efficient secure multiparty computation (MPC) protocols for securely evaluating Boolean circuits where XOR gates are considerably cheaper (no communication, less computation) than AND gates. There are also many MPC protocols where each AND gate of the evaluated circuit requires interaction and so the performance depends on both the multiplicative complexity (MC) and ANDdepth of the circuit. Examples are the semi-honest secure version of the GMW protocol [37], and tiny-OT [52] with security against malicious adversaries. Applications of symmetric encryption schemes in these protocols include privacy-preserving keyword search based on Oblivious Pseudorandom Functions (OPRFs) [33], set intersection [42] and secure database join [47]. More details to motivate the use of symmetric encryption in MPC are given in [3].

1.2 The design strategy Rasta and its background

In this paper, we propose a design strategy called Rasta[5] for symmetric encryption that simultaneously achieves very low values in two of the three considered metrics: Symmetric encryption that has ANDdepth d and at the same time only needs d ANDs per encrypted bit. The main result is that even for very low $d = 2, \dots, 6$, we can give some evidence that attacks may not exist.

We achieve this by putting so-called ASASA-like permutation constructions into a new setting. Generic substitution-permutation designs which interleave a key-dependent affine layer with key-dependent S-boxes have been studied since SASAS [14]. Follow-up work refined and extended this line of inquiry, and put it also into use for the purpose of white-box implementations of symmetric ciphers, and for instantiating schemes with public-key-like properties [12].

Our *new twist* to ASASA-like constructions is to consider a setting where the substitution layer is suitably chosen, public and fixed, but the affine layers are *derived from a public nonce and a counter* such that no affine layer is likely to be ever re-used under a single key (see Fig. 1). This approach prevents the attacks [28,36,50] that broke the proposals of [12], as an adversary will never be able to query the same ASASA-like permutation more than once in Rasta. Since the setup of each instance via the extendable-output function (XOF) [51] depends only on public information (N, i), it does not contribute to the (homomorphic) circuit evaluation cost in applications like FHE. In addition to the number of rounds, we also consider key sizes (and so block sizes of the used permutation) that are bigger than the required security level as tunable security parameter to provide protection against certain attack vectors.

Variants. The practical downside of Rasta with a very low d is that for a fixed security level, the required key size and the number of additions needed for its evaluation grows very fast, see also the comparison in Table 1. Hence we consider such parameters as non-practical and at times use gray coloring in tables or figures for it. Throughout the paper we describe ways to bound

[5] The name Rasta originates from the use of randomly looking affine layers A and the repetition of affine and S-box layer $(\mathrm{AS})^*$ followed by a last affine layer A. In short $\mathrm{R(AS)}^*\mathrm{A}$. A C++ reference implementation is available at: `https://github.com/iaikkrypto/rasta`.

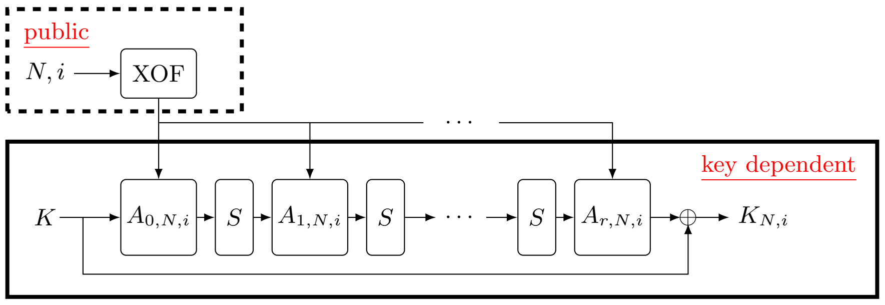

Fig. 1. The r-round Rasta construction to generate the keystream $K_{N,i}$ for block i under nonce N with affine layers $A_{j,N,i}$.

various classes of attacks and use them to derive key and block sizes for any depth d. However, we do not have attacks matching these bounds. As we can see in Sect. 3, the attacks we have are rather far away from these bounds. In order to also explore the limits of what this design approach might achieve and to further encurage more cryptanalysis we also propose a variant of Rasta called Agrasta where the key and block size equals the security level (plus one to get odd numbers) and basing the number of rounds on what we can attack plus a security margin. Fig. 2 brings the area where we know attacks in relation to the instances of Rasta and Agrasta having 80-bit security. Note that the area is mostly defined by cases, where the maximal number of different monomials becomes so low, that the equation system can be solved by a trivial linearization.

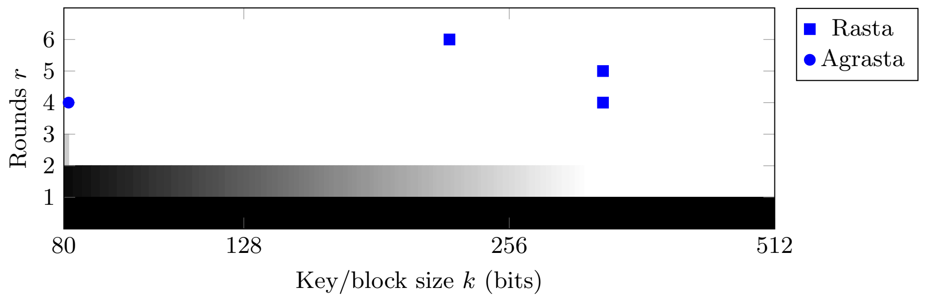

Fig. 2. Security margin of Rasta and Agrasta instances having 80-bit security. White area cannot be attacked with a complexity less than 2^{80}. Black area can be attacked with complexity below 2^{50}.

1.3 Related work and comparison

Recently, a number of new primitives were proposed that aim to minimize metrics related to the computation of AND gates. For a conjectured security of 128 bits, LowMC and Kreyvium require an ANDdepth of 11 or more, whereas FLIP manages to have a much lower ANDdepth of 4. The total number of AND computations is however much larger in FLIP (1072 per bit, 112 from the quadratic function, and 960 from the triangular function) than in LowMC or Kreyvium (which can be as low as 3 to 4 ANDs per bit). We give a more detailed discussion and comparison of them in the following. MiMC [2] is a very different design that shows excellent properties in a broad range of use-cases incl. MPC and SNARKs. It's main feature is that it can operate on elements in GF(p) natively and as such is very different to all the other designs we consider in our comparison.

High-level approach. What we do in this approach is to make a significant part of the computations independent of the key. This high-level approach was (perhaps for the first time) used in the FLIP design [49]. While in FLIP it is only the key bits that are permuted in a nonce-dependent way and the rest of the construction is fixed, in our design many essential parts of the construction are nonce-dependent: The derivation of a suitable affine layer, for every block, based on nonce and counter inputs using an extendable-output function (XOF) [51].

The advantage of this idea is that operations which do not depend on the key are in various settings of interest much less costly than operations that do depend on the key. In our experimental validation of the proposed approach, we include in the runtime the construction of each affine layer.

This results in a nice advantage of our approach as it allows for security arguments in the case of outputting many more than a single bit, hence drastically improving the number of ANDs per bit. Note that FLIP mitigates this property by focusing on a class of homomorphic encryption schemes where error growth is quasi-additive when considering a multiplicative chain and hence the large number of AND computations per encrypted bit are less of an issue.

New cryptanalytic insights. As a side-effect of these novel designs, new and interesting cryptanalytic insights continue to emerge. Attacks on earlier versions of LowMC [4] led to new insights on how higher-order properties can get extended because of non-full S-box layers [29,31] and novel optimization of interpolation attacks [29]. As a result, the LowMC v2 parameters are larger: For 80-bit security at least 12 instead of 11 rounds are needed, and for 128-bit security at least 14 instead of 12 rounds are needed. The 12-round version was shown to offer less than 128-bit of security. In this paper we consider both versions, because comparisons in the past have been done with v1. Another example are attacks on FLIP which showed that guess-and-determine attacks [32] force designers to choose more conservative parameters for their novel design.

Comparison with respect to AND-related metrics. For a security level of 128 bits, LowMCv2 has a depth of at least 14, Kreyvium [20] has depth of at least 12 and the most recent proposal FLIP [49] only needs a depth of 4. The comparison among these three custom designs is however more complicated than these numbers might suggest. Whereas for LowMC the depth remains constant for the encryption of at least 256 bits, for Kreyvium the depth starts to grow after 67 bits already. The very low depth of FLIP comes at the cost of a much larger number of AND computations per encrypted bit: At a security level of 128 bits it is 1072 ANDs/bit for FLIP compared to values as low as 3 to 4 for Kreyvium and LowMC.

Table 1 and Fig. 3 illustrate a comparison of our design with these three earlier designs. They overlap partially in the content they convey, but also complement each other. For simplicity, in Fig. 3, we only show the figure for the security level of 128 bits, also because only for this particular security level, instantiation proposals are available for all design options.

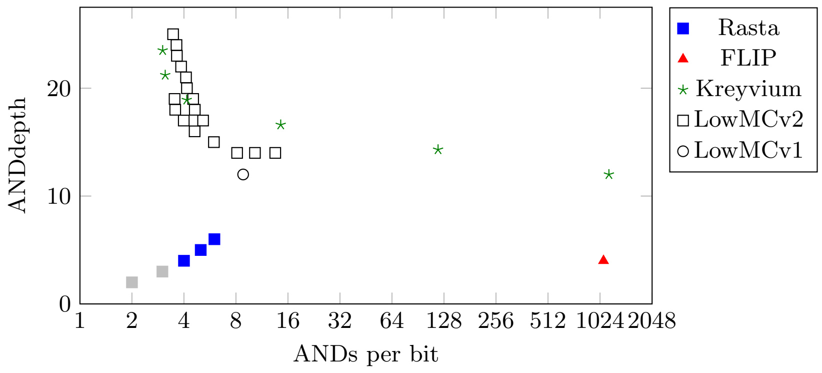

Fig. 3. Comparison with respect to the two most important metrics. All are for a security level of 128 bits. The different points for Kreyvium are derived from varying the number of output bits generated per initialization. For LowMCv2 the given round formula is used to explore various possible trade-offs in the design space.

Fig. 3 does not only give single data points for LowMC and Kreyvium, but explores a wider range of usable options. As the stream cipher Kreyvium needs an initialization phase, the number of ANDs/bit is very high if only a small number of keystream bits are generated. The more keystream bits are generated, the more this initialization phase is amortized and hence reduces the number of ANDs/bit, but on the other hand the ANDdepth is growing.

LowMC does not refer to a particular block cipher geometry, but allows to generate instantiations for a wide range of block sizes, security levels, and number of S-boxes per round. For Fig. 3, we fixed the security level to 128-bit, but tried a number of block sizes in the range between 256 and 1024 bits for two different choices of number of S-boxes per round (33 and 63), simply to get an impression of the trade-space between the two metrics.

What is however not visible in Fig. 3 are two other properties these schemes have: key size and block size. This is shown, together with all other metrics in Table 1, for 80-bit, 128-bit and 256-bit security levels. Whereas Kreyvium and

Table 1. Comparison of Rasta with related designs. ℓ is the number of encrypted bits. For Rasta and LowMC, ℓ needs to be a multiple of the block size.

Name	security	key size	block size	ANDdepth	minAND	ANDs per bit
LowMCv1 [4]	80	80	256	11	1617	6.3
LowMCv2 [3]	80	80	256	12	1764	6.89
LowMCv2 [3]	80	80	128	12	1116	8.72
Trivium [26]	80	80	1	$12 + \log(\ell)$	$1152 + 3 \cdot \ell$	3–1152
FLIP [49]	80	530	1	4	352	352
Rasta	80	219	219	6	$6 \cdot \ell$	6
Rasta	80	327	327	5	$5 \cdot \ell$	5
Rasta	80	327	327	4	$4 \cdot \ell$	4
Rasta	80	3939	3939	3	$3 \cdot \ell$	3
Rasta	80	$\approx 2^{21}$	$\approx 2^{21}$	2	$2 \cdot \ell$	2
LowMCv1 [4]	118	128	256	12	2268	8.86
LowMCv2 [3]	128	128	256	14	2646	10.34
LowMCv2 [3]	128	128	192	14	2592	13.50
Kreyvium [20]	128	128	1	$12 + \log(\ell)$	$1152 + 3 \cdot \ell$	3–1152
FLIP [49]	128	1394	1	4	1072	1072
Rasta	128	351	351	**6**	$\mathbf{6} \cdot \ell$	**6**
Rasta	128	525	525	**5**	$\mathbf{5} \cdot \ell$	**5**
Rasta	128	1877	1877	**4**	$\mathbf{4} \cdot \ell$	**4**
Rasta	128	$\approx 2^{18}$	$\approx 2^{18}$	**3**	$\mathbf{3} \cdot \ell$	**3**
Rasta	128	$\approx 2^{33}$	$\approx 2^{33}$	**2**	$\mathbf{2} \cdot \ell$	**2**
LowMCv2 [3]	256	256	512	18	3564	6.96
Rasta	256	703	703	**6**	$\mathbf{6} \cdot \mathbf{l}$	**6**
Rasta	256	3545	3545	**5**	$\mathbf{5} \cdot \ell$	**5**
Rasta	256	$\approx 2^{19}$	$\approx 2^{19}$	**4**	$\mathbf{4} \cdot \ell$	**4**
Rasta	256	$\approx 2^{34}$	$\approx 2^{34}$	**3**	$\mathbf{3} \cdot \ell$	**3**
Rasta	256	$\approx 2^{65}$	$\approx 2^{65}$	**2**	$\mathbf{2} \cdot \ell$	**2**

LowMC allow for keys as short as the security level in bits, both FLIP and Rasta require longer keys. Rasta, similarly to LowMC, requires a much larger number of XOR operations than FLIP or Kreyvium. Whereas traditionally the cost of linear operations is considered almost negligible compared to non-linear operations, the number of XOR operations is so extreme in the setting we are interested in that it is no longer negligible. Hence as we will see in the validation of the practical usefulness in Sect. 4, more moderate choices of d between 4 and 6 seem more useful.

Implementation comparisons. Even though the main contribution of this work is the analysis of a scheme that has at the same time very low ANDdepth and ANDs/bit, we also aim to validate the design approach by means of actual implementations. As discussed already, the practical downside is that for a fixed security level and for a very low d (meaning very low ANDdepth and ANDs/bit), the required key size and especially the number of additions needed for its evaluation grows very fast. Hence the question is: can variants of Rasta be useful in practice?

It turns out that implementations of our scheme with too low d are not possible. For some parameters this is already obvious from the huge required key and block size. As we will show in Sect. 4 also more moderate sizes can be prohibitive in the FHE setting we use as a test-case. Nevertheless, we also give some evidence that for more moderate choices of d between 4 and 6, the resulting cost of homomorphically evaluating the circuit of our new construction may well be competitive.

1.4 External cryptanlysis

We are already aware of cryptanalysis attempts outside the design team. This includes work by Raddum on a dedicated attack [54] as well work by Bile, Perret and Faugère [11] on algebraic and Gröbner bases approaches.

1.5 Organization of the paper

In Sect. 2, we give the detailed specification of the new design approach together with concrete parameters for instantiations of Rasta at various conjectured security levels. In Sect. 3, we thoroughly analyze the security of Rasta, considering both standard symmetric cryptanalysis techniques and some novel, design-specific attack angles. In Sect. 4, we discuss concrete implementations and provide a performance comparison with other designs. Finally, we conclude and discuss open problems in Sect. 5.

2 Specification

2.1 Encryption mode

Rasta is a family of stream ciphers. To produce the keystream, it applies a permutation with feed-forward, as shown in Fig. 4. The input of the permutation is the secret key K, where the key size k matches the block size n of the underlying family of permutations $P_{N,i}$. The keystream is generated by using different permutations $P_{N,i}$ per encrypted block, which are parametrized by the nonce N and a block counter i. To ensure confidentiality, a new and unique nonce N is required for every encryption.

2.2 Permutation $P_{N,i}$

Rasta's family of permutations $P_{N,i}$ applies r rounds of different affine layers $A_{j,N,i}$ and non-linear layers S. After r rounds, a final affine layer is applied:

$$P_{N,i} = A_{r,N,i} \circ S \circ A_{r-1,N,i} \circ \cdots \circ S \circ A_{1,N,i} \circ S \circ A_{0,N,i}$$

Each affine layer is different and depends on the nonce N and block counter i. The family of permutations is parameterizable in the number of rounds r and permutation block size in bits n, where n is an odd number.

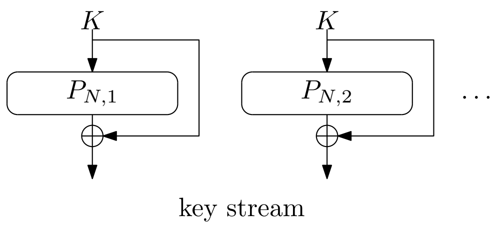

Fig. 4. Encryption mode of Rasta.

Non-linear layer S. For the non-linear layer, we apply the χ-transformation [25, Section 6.6.3], previously used in Keccak [8] and ASASA [12], to the entire block. This transformation is invertible for any odd number of bits n [25]. For input bits x_i and output bits y_i, $0 \leq i < n$, the non-linear layer is defined in (1), with all indices modulo n:

$$y_\ell = x_\ell \oplus x_{\ell+2} \oplus x_{\ell+1}x_{\ell+2} \,. \tag{1}$$

Affine layers $A_{j,N,i}$. The affine layers are a simple binary multiplication of a binary $n \times n$ matrix $M_{j,N,i}$ to the input vector x, followed by the addition of a round constant $c_{j,N,i}$:

$$y = M_{j,N,i} \cdot x \oplus c_{j,N,i} \,.$$

Generation of matrices $M_{j,N,i}$ and round constants $c_{j,N,i}$. We propose to generate the different matrices and round constants with the help of an extendable-output function (XOF) [51] that is seeded with the number of rounds r, block size n, nonce N, and i. Hereby, the output of the XOF is first used to generate $M_{0,N,i}$, a unstructured nonsingular Matrix. Several ways are possible to generate such matrices with the help of an XOF output. The first one is to add rows and check if the matrix is invertible. As pointed out by Randall [55], on average it takes three tries before we end up with a nonsingular matrix. In the same paper, Randall [55] gives an algorithm to generate random nonsingular $n \times n$ matrices needing less than $n^2 + 3$ random bits inserted in a clever way in two $n \times n$ matrices, which are multiplied in the end. The choice of the algorithm to generate nonsingular matrices should not influence the security, just the execution time. However, communicating instances have to use the same algorithm.

After the generation of the first matrix, the output of the XOF is used to create $c_{0,N,i}$, $M_{1,N,i}$, and so on. To ensure that the permutation is secure, we expect the XOF to behave like a random oracle up to a certain security level. For instance, it should not be feasible for an attacker to find inputs to the XOF for outputs of the attacker's choice except by repeatedly querying the XOF for different inputs. Furthermore, the internal state of the XOF should be large enough so that internal collisions within its state are prohibited. A suitable choice for an XOF would be for most instances SHAKE256 [51].

2.3 Design rationale

The essential idea of Rasta is to reduce the ANDdepth as much as possible by creating a moving target, which is only evaluated once per key. Hence, we have decided to use a permutation with feed-forward, where the secret occupies the whole input and the keystream is generated by always evaluating a different permutation. Those permutations are obtained by choosing new matrices and round constants for each new permutation call, based on an XOF seeded with a public nonce and block counter. This technique allows us to treat matrices and round constants during our analysis as if they where randomly created for each different permutation (new nonce and counter pair), with the restriction that same nonce and counter pairs give us always the same permutation and hence, the same matrices and round constants. Since the XOF uses no secret, it can be publicly evaluated and thus, similar as for FLIP [49], the XOF does not influence the AND related metrics.

Choosing the matrices this way minimizes structural similarities between and within the permutation instances. The round constants remove obvious fixed-points, such as the fixed-point 0 that all instances would have in common otherwise. Furthermore, these round constants add an additional layer of protection against attacks between single instances of the permutation. The affine layer is required to be a permutation rather than a function to prevent reduction of the state space. Otherwise, if for example the final affine layer did not have full rank, then the final key feed-forward would allow an attacker to derive information about a linear subspace containing the key with each query.

Instead of smaller S-boxes for the non-linear layer, we use one large χ-transformation [25] across the entire state. Since we only need to evaluate it in forward direction, we benefit from the fact that it is very efficient to evaluate in forward direction with a degree of only 2, while its inverse has a very high degree of $(t + 1)$ for size $(2t + 1)$ [12].

2.4 Instances

Based on the security analysis done in Sect. 3, we propose block sizes for 4 to 6 rounds of Rasta aiming at 80, 128, or 256 bits of security in Table 2. These block sizes are derived from the results of our security analysis shown in Table 5 and Table 6 in Sect. 3. Since part of our analysis relies on good diffusion properties, we only recommend instances of Rasta with at least 4 rounds for use. However, from a theoretical perspective, smaller instances are also of interest and hence, we add parameters for 2 and 3 rounds (in gray) to Table 2 as well.

Data limit and related-key attacks. Our goal is to derive instances that have both a small block size and a small ANDdepth. To make this feasible, we limit the data complexity per key to $\sqrt{2^s}/n$ blocks, where s is the targeted security level in bits and n is the block size. Furthermore, to ensure low ANDdepth with our construction in general, related-key attacks are out of scope.

Table 2. Minimal block sizes for 4 to 6 rounds of Rasta aiming to provide 80, 128, or 256 bits of security (2 and 3 rounds in gray).

Security level	Rounds				
	2	3	4	5	6
80-bit	2 320 961	3 939	327	327	219
128-bit	9 506 325 433	246 831	1 877	525	351
256-bit	40 829 356 287 426 864 861	16 167 762 975	445 939	3 545	703

Agrasta. As already mentioned, we have chosen the block sizes for Rasta in a conservative manner based on our security evaluation in Sect. 3. The block sizes of Rasta are mostly determined by the bounds we get on the maximal number of monomials and the bounds on the probability that good linear approximations exists. However, this does not mean that we have matching attacks for these bounds, in fact, as can be seen in Sect. 3, what we actually can attack is far away from these bounds. Hence, we propose Agrasta shown in Table 3.

Table 3. Instances of Agrasta.

Security level	Rounds	Blocksize
80-bit	4	81
128-bit	4	129
256-bit	5	257

Agrasta has a block size that matches the security level (or block size plus 1 for even security levels). For the number of rounds, we consider a minimal number of 4 rounds for the same reasons as for Rasta and add rounds until we cannot attack the construction anymore. As a consequence, Agrasta has a block size of 81-bit for 80-bit security having 4 rounds, 129-bit for 128-bit security having 4 rounds and 257-bits for 256-bit security having 5 rounds (in this case trivial linarization would work for 4 rounds).

Toy version. To encourage cryptanalysis of Rasta, we specify toy versions of Rasta in Table 4. Those toy versions aim at achieving 24-bit security.

Table 4. Toy versions of Rasta.

Security level	Rounds				
	2	3	4	5	6
24-bit	193	193	97	97	65

3 Security analysis

In this section, we discuss various cryptanalytic approaches and argue the security of the recommended parameter configurations of Rasta. We focus on key recovery attacks and assume that the attacker can obtain the keystream $K \oplus P_{N,i}(K)$ for arbitrary choices of (N, i). In particular, assume the attacker requires instances with a particular property of the affine layers that occurs with probability p. This chosen-nonce setting allows the attacker to obtain keystream for such an instance with about p^{-1} XOF queries (in key-independent precomputation) and 1 Rasta query, instead of p^{-1} Rasta queries. We start out with algebraic attacks via linearization and Gröbner bases, then we describe a potential attack vector via linear approximations and argue why various classical attacks such as differential, integral, or higher order attacks are ruled out or unlikely. In Section 3.4 we then briefly describe a dedicated attack on variants that are very close to Agrasta but reduced to three rounds. Last but not least we describe various experiments on toy versions of Rasta, incl. SAT solver, Gröberner-bases, and an analysis of linear properties and monomial counts. A discussion on the security of variants without the first or last affine layer can be found in our Appendix C.

3.1 Algebraic attacks

We first consider algebraic attacks that aim to recover the secret key K by solving a system of non-linear Boolean equations. These equations are collected by observing the keystream for different nonces, and setting up the algebraic normal form (ANF) for each observed instance of the permutation. All these ANF polynomials share the bits of K as unknowns. In the following, we consider different approaches and trade-offs to solve these equations.

Trivial linearization. In this attack, the resulting non-linear system of equations is solved by substituting the non-linear terms with new variables. Consider a cipher with algebraic degree ϕ and a key of k bits. For such a cipher, the number of unknowns in our equations is upper-bounded by k^{ϕ}. Thus, after collecting at most k^{ϕ} linearly independent equations, the system can definitely be solved and the key recovered. The maximum number of different monomials we can get is:

$$U = \sum_{i=0}^{\phi} \binom{k}{i}. \qquad (2)$$

For instance, considering Rasta with one S-layer of degree 2 and a key of 1024 bits, we need at most 2^{19} equations. For degree 16 and a 1024-bit key, we already get up to $2^{115.6}$ monomials, and for degree 16 with a 2048-bit key, $2^{131.68}$ monomials. Since the ANDdepth d of the cipher is bound to the degree ϕ ($d = \log_2 \phi$), we have an immediate impact on reasonable ANDdepths.

Key-guessing to reduce monomials. Guessing g out of k key bits reduces the number U of possible monomials and hence the number of variables we need to introduce to linearize and solve the system to

$$U = \sum_{i=0}^{\phi} \binom{k-g}{i}. \tag{3}$$

Guessing g bits of the key reduces the data needed to perform a linearization attack. However, the linear system has to be solved 2^g times for every key guess, giving a maximum number of bits corresponding to the security level of the scheme that can be guessed.

On the number of monomials. While the total number of monomials we can get gives us insight when the system of equations can definitely be solved, the number of monomials we get per output equation plays an important role in algebraic attacks. For instance, getting too many sparse equations might lead to a sub-system that can be solved or might enable other attacks similar to algebraic or fast algebraic attacks on stream ciphers [23,24]. Therefore, we study the average number of monomials after r rounds of Rasta, and we compare it with the worst-case number. We conclude that the number of monomials in the average case is well approximated by the worst case.

We first consider the worst case, which was already studied in the previous section. Since the S-layer has degree 2, the degree of the scheme after r rounds is 2^r, so the number of monomials is upper-bounded by $\sum_{i=0}^{2^r} \binom{k}{i}$.

To analyze the average case, recall that the matrices A of the linear layers are uniformly distributed. Consider one round $S \circ A(x)$ of Rasta with input $x = (x_0, \ldots, x_{k-1})$, and let $A(x) = A \cdot x + c$:

$$S \circ A(x)_i = \bigoplus_{j=0}^{k-1} \bigoplus_{l=j+1}^{k-1} a^i_{j,l} \cdot x_j \cdot x_l \oplus \bigoplus_{j=0}^{k-1} b^i_j \cdot x_j \oplus g^i,$$

where remember that $x^2 = x$ for each $x \in \mathbb{F}_2$ and where

$$\begin{aligned}
a^i_{j,l} &= A_{i+1,j} \cdot A_{i+2,l} \oplus A_{i+2,j} \cdot A_{i+1,l},\\
b^i_j &= A_{i,j} \oplus c_{i+2} \cdot A_{i+1,j} \oplus (1 \oplus c_{i+1}) \cdot A_{i+2,j},\\
g^i &= c_i \oplus c_{i+2} \oplus c_{i+1} \cdot c_{i+2}.
\end{aligned}$$

First, we focus on the monomials of degree 2. The probability that a coefficient $a^i_{j,l}$ is equal to 0 is $5/8$:

$$\begin{aligned}
\mathbb{P}[a^i_{j,l} = 0] &= \mathbb{P}[A_{i+1,j}A_{i+2,l} = A_{i+2,j}A_{i+1,l} = 0] + \mathbb{P}[A_{i+1,j}A_{i+2,l} = A_{i+2,j}A_{i+1,l} = 1]\\
&= \left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2 = \frac{5}{8}.
\end{aligned}$$

Thus, the probability that all the coefficients of the variable $x_j \cdot x_l$ with $l > j$ are equal to 0 is

$$\mathbb{P}[a^i_{j,l} = 0 \quad \forall i = 0, \ldots, k-1] = \left(\frac{5}{8}\right)^k,$$

or equivalently, at least one of these coefficients is different from 0 with probability $1 - (5/8)^k$. Since this is true for each i, we expect an average number of monomials of degree 2 equals to

$$\binom{k}{2} \cdot \left[1 - \left(\frac{5}{8}\right)^k\right] \simeq \binom{k}{2},$$

where the approximation holds for $k \gg 1$. In a similar way, we expect an average number of monomials of degree 1 equal to $k \cdot (1 - 2^{-k}) \simeq k$. It follows that the average number of monomials is well approximated by the worst one[6]. Our experiments confirm this prediction: we observed that all the monomials of degree 1 and 2 are present in the system made by the equations of the output bits of $S \circ A(x)$. Since the same argumentation holds for the following rounds, this justifies our claim.

Another consideration that strengthens our claim is the following. Suppose that the average number v of variables is much lower than the number w of variables in the worst case, that is, $v \ll w$. Thus, given one encryption, the number of variables that one has to consider is only v. However, in order to find a solution for the given system of linear equations, one must consider other encryptions. Due to the previous hypothesis, for each one of these new encryptions, one expects an average number of v variables. On the other hand, since the linear layers change for each encryption, it is not possible to claim that the variables of these texts are always the same. In other words, the variables of a second encryption are in general different from the ones of the first encryption. This implies that if one uses a second encryption to find the solutions of the linear equations, the number of variables that one has to consider is on average not v, but greater than v (and lower than $2 \cdot v$). Due to the same argumentation, using r texts, this number is on average (much) greater than v but lower than $r \cdot v$. Intuitively, if r is big enough, even if for each single text/encryption the number of variables v is lower than the total one (i.e. w), using r texts to find the real values of these variables implies that this number is closer to w than to v. As a consequence, this provides another intuitive reason while one has to consider the worst number of variables in order to evaluate the security of this proposed encryption scheme.

Gröbner basis computation. Instead of linearization, one could also try to solve the non-linear system by computing a Gröbner basis for the system of

[6] Note here that an attacker could rename the variables obtained after the linear layer, making the number of quadratic terms drops to only k. However, this renaming would only be valid for one message, while many messages are necessary to solve the system, which makes this process unlikely to have any impact.

equations. However, it is highly unlikely that this attack vector threatens our construction. The main point is that the number of unknowns is very large (starting from 219-bit blocks and keys) and the algebraic degree is not that small (starting from 2^4). This has to be compared with the best results to date, that to the best of our knowledge allow to solve a system with up to 148 *quadratic* equations in 74 variables [43]. Actually, a recent technical report [11] provides further evidence that solving the corresponding system of equations for Rasta becomes quickly infeasible.

Discussion. As we have seen in this section, Rasta gives us a system of non-linear Boolean equations of a certain degree ϕ with dense equations on average. This system of equations depends on the two parameters of Rasta, the block size and the ANDdepth. We have plotted the maximum number of different monomials under guessing 80-bit, 128-bit, 256-bit key information for various block sizes and ANDdepths in Fig. 5.

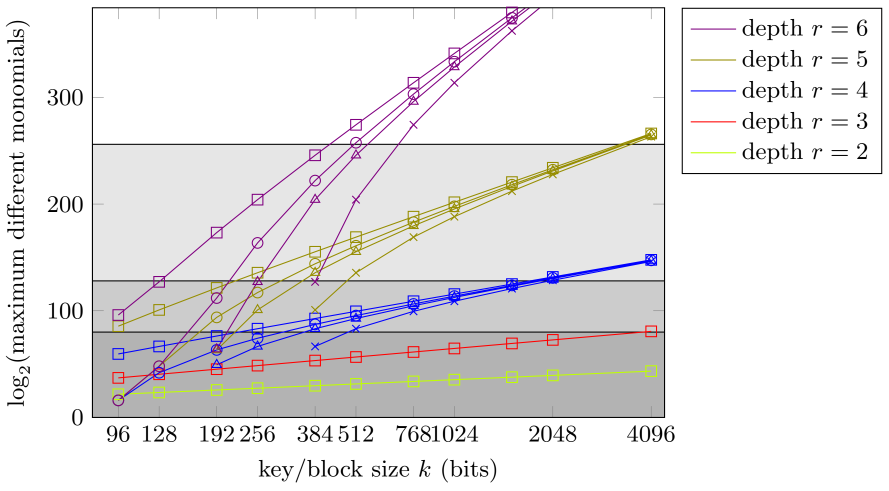

Fig. 5. Trivial linearizion attack. □/○/△/× denote 0/80/128/256-bit key guess.

This is particularly interesting, since Fig. 5 gives us insight in the data an attacker has to acquire before the system can be definitely solved by using trivial linearization. At the moment, the costs of solving a linear system of U linear equations in U binary variables is about $\mathcal{O}(U^\omega)$ bit operations, where ω is bigger than 2. Nevertheless, the key can still be recovered with k equations in $\mathcal{O}(2^k)$ using brute-force.

Estimating the required time complexity for solving a non-linear system with $k+\epsilon$ equations is an open research question in the case of Rasta. Since we want to limit the possibilities an attacker has to solve the system of non-linear equations,

Table 5. Block sizes such that the number of monomials is greater than 2^s.

Security level s	Depth				
	2	3	4	5	6
80-bit	2 320 961	3 938	305	167	161
128-bit	9 506 325 433	246 831	1 876	348	258
256-bit	40 829 356 287 426 864 861	16 167 762 975	445 938	3 545	682

we set a limit on the number of equations acquirable by the attacker. Therefore, we apply the two following restrictions for a security level of s bits, which will influence the block and key size of Rasta:

- Choose key size k and degree ϕ such that $\sum_{i=0}^{\phi}\binom{k-g}{i} > 2^s$ (Table 5).
- Data complexity limit of $\sqrt{2^s}/k$ blocks.

3.2 Attacks based on linear approximations

In this section, we deal with classes of attacks that exploit linear approximations having a high bias δ.[7] Although classical linear cryptanalysis [48] seems not to be applicable to Rasta in a straight forward manner, since it would require the repeated evaluation of the same Rasta permutation with different inputs, other attacks based on linear approximations can still be a threat. For instance, in the case of Rasta, an attacker can collect single evaluations of different linear approximations that are valid with a certain probability. Here, the attacker faces a problem similar to the LPN-problem and thus, algorithms similar to existing ones designed for solving the LPN-problem [15] might be applicable.

However, attacks utilizing linear approximations have in common that the data-complexity required to perform these attacks is bound to the bias δ of the used linear approximations and usually lies in regions of δ^{-2}. Therefore, attacks exploiting linear approximations are not applicable if linear approximations that have a good bias do not exist. Hence, we want to bound the probability that a certain choice of the nonce N gives us matrices that allow linear approximations with a good bias δ.

Bounds for 2 S-layers (depth 2). First, we restrict our observations on two S-layers with one matrix M_1 in between. If we restrict our observations just to the matrix M_1, we can make the following statements about the quality for a certain linear characteristic, where a_0 bits are active at the input of M_1 and a_1 bits are active at the output.

As already shown by Daemen [25], the correlation weight $\mathrm{w_c}$ of a linear approximation for the χ-transformation only depends on the active bits of the

[7] We do not extend this analysis to linear hulls as even for more suitable designs doing this analytically is beyond the state of the art.

output of the χ-transformation and its correlation is either zero or $2^{-\mathrm{w_c}}$. As noted in [7], the correlation weight is equal to half the number of active bits plus a positive number that depends on the position of the active bits. If all bits of the output are active, then the correlation weight is half the block length minus 1. Since in our case the block length is always at least the security level, we can ignore this special case and can upper-bound the bias just relying on half the number of active bits at the output of the S-layers.

However, as mentioned in the beginning of the section, we want to make statements on a linear characteristic just using the information that a_0 bits are active at the output of the first S-layer and a_1 bits are active on the input a second one. Thus, we have to make assumptions on the minimum number of active output bits, dependent on the number of active input bits. Let us assume that just one bit on the output of an S-layer is active. How many bits at the input can be active, so that we get a correlation, or bias different from zero? As we can see from (1), in order to determine the value of the active bit, just 3 bits of the input are needed. Hence, just these 3 bits can be used in a linear approximation of the output bit, while trying to include any other bit in a linear approximation definitely results in a bias of zero. So we know that if we have 3 active bits at the input of the S-layer, at least one output bit has to be active, while for 4 or more active input bits definitely 2, or more outputs have to be active. Considering two active output bits, at most 6 bits are used in their calculations, which in turn can be used in a linear approximation and so on. Therefore, a linear approximation using x active bits at the output might use less than $3x$ input bits, but never more.

Thus, we can upper-bound the correlation of a resulting linear approximation with $2^{-(a_0+\lceil a_1/3\rceil+1)/2}$ and hence, the bias with $2^{-(a_0+\lceil a_1/3\rceil)/2-1}$. So, our goal is to make statements about the probability that a randomly generated matrix M_1 allows linear characteristics with a_0 active bits at the input and a_1 active bits at the output, so that $(a_0 + \lceil a_1/3\rceil)$ is smaller than a certain threshold. Consider that in Sect. 3.1, we already introduce a limit on the data complexity, we assume that linear approximations with a bias of $2^{-s/4}$ cannot be exploited in attacks, where s is the security level in bits. As shown in the Appendix the number of invertible matrices that allow a certain fixed linear characteristic is $\prod_{i=1}^{n-1}(2^n - 2^i)$. This number is independent of the concrete pattern. By counting the number of suitable linear characteristics, we get:

$$\begin{aligned}
\mathbb{P}_2\left[-\log_2(\delta) \le \frac{s}{2}\right] &\le \sum_{\substack{0<a_0,a_1\\(a_0+\lceil a_1/3\rceil)\le s/2}} \binom{n}{a_0}\binom{n}{a_1}\frac{\prod_{i=1}^{n-1}(2^n-2^i)}{\prod_{i=0}^{n-1}(2^n-2^i)}\\
&\le (2^n-1)^{-1}\sum_{\substack{0<a_0,a_1\\(a_0+\lceil a_1/3\rceil)\le s/2}} \binom{n}{a_0}\binom{n}{a_1}\\
&\le (2^n-1)^{-1}\sum_{\substack{0\le a_0,a_1\\(a_0+a_1)\le \frac{3s}{2}}} \binom{n}{a_0}\binom{n}{a_1} \le (2^n-1)^{-1}\sum_{0\le a\le \frac{3s}{2}} \binom{2n}{a}.
\end{aligned}$$

$$M_0 \longrightarrow \underbrace{S_0 \longrightarrow M_1 \longrightarrow S_1}_{P_1} \longrightarrow M_2 \longrightarrow \underbrace{S_2 \longrightarrow M_3 \longrightarrow S_3}_{P_2} \longrightarrow M_4$$

Fig. 6. Schematic of the Rasta permutations having 4 S-layers.

Bounds for 4 S-layers (depth 4). Fig. 6 shows the Rasta permutation having 4 S-layers. We will show bounds on the probability that randomly generated matrices allow linear characteristics having a bias higher than some threshold. As indicated in Fig. 6, we will calculate the probabilities that randomly generated matrices allowing a certain number of active bits independently for M_1 and M_3, assuming that every randomly generated matrix M_2 is able to connect the resulting linear characteristics.

As already mentioned, we do not want to have linear characteristics with a bias higher than $2^{-s/4}$. Hence, the probability that a matrix M_1 allows transitions with a_0 and a_1 active bits and that a matrix M_3 allows transitions with a_2 and a_3 active bits, where $(a_0 + \lceil a_1/3 \rceil + a_2 + \lceil a_3/3 \rceil) \leq s/2$, should be small. We can calculate this probability by summing up the probabilities for fixed a_0, a_1, a_2, and a_3 up to this bound:

$$\begin{aligned} \mathbb{P}_4 \left[-\log_2(\delta) \leq \frac{s}{2} \right] &\leq (2^n - 1)^{-2} \times \sum_{\substack{0 < a_0, a_1, a_2, a_3 \\ (a_0 + \lceil a_1/3 \rceil + a_2 + \lceil a_3/3 \rceil) \leq \frac{s}{2}}} \binom{n}{a_0} \binom{n}{a_1} \binom{n}{a_2} \binom{n}{a_3} \\ &\leq (2^n - 1)^{-2} \times \sum_{0 \leq a \leq \frac{3s}{2}} \binom{4n}{a} \end{aligned} \tag{4}$$

Bounds for r S-layers (depth r). Finally, we can extend the concept we have used for bounding the probability on 4 rounds to bounding the probability for an arbitrary even number r of rounds of Rasta:

$$\mathbb{P}_r \left[-\log_2(\delta) \leq \frac{s}{2} \right] \leq (2^n - 1)^{-r/2} \times \sum_{0 \leq a \leq \frac{3s}{2}} \binom{r \cdot n}{a} \tag{5}$$

Discussion. We visualize the effects of different block sizes and depths on the probability that a linear approximation for such an instance of Rasta exists in Fig. 7. If we compare 80-bit security for 2 rounds of Rasta with 80-bit security for 4 rounds in Fig. 7, we see that we get similar probabilities for 4 rounds of Rasta, requiring just half the block size compared to two rounds. This gets already clear from equation (5), where we see that—at least for our bounds—doubling the depth has similar effects as doubling the block size.

To give a better overview about the influence of linear approximations on the block size, we summarize reasonable block sizes for different security levels and depths in Table 6. Concretely, we do not want to have linear approximations

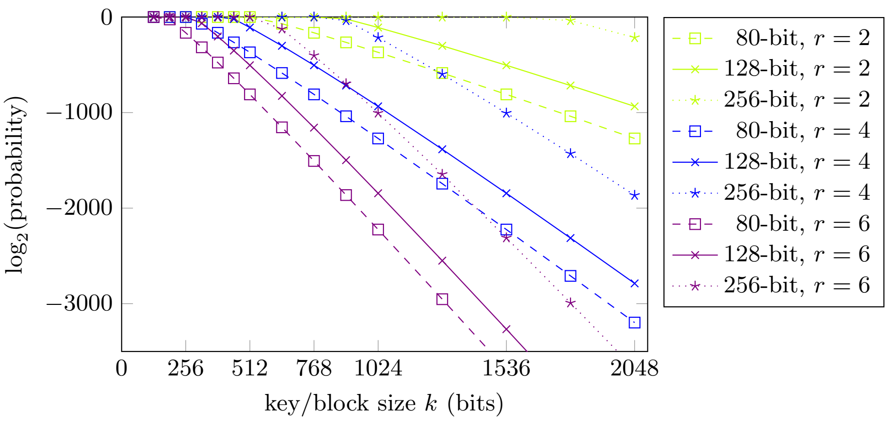

Fig. 7. Probability that a linear trail/approximation with bias above for a certain security level exists.

Table 6. Block sizes where $-\log_2(\mathbb{P})$ matches s.

Security level s	$r = 2$	$r = 4$	$r = 6$
80-bit	654	327	218
128-bit	1050	525	350
256-bit	2106	1053	702

where the squared bias δ^2 matches our data limit. Hence, the block sizes for this table have been chosen so that the probability that a linear approximation with a bias $\geq 2^{-s/4}$ exists (for one random choice of the linear layer) exceeds 2^{-s}, where s is the security level. Therefore, for block sizes exceeding the ones given in Table 6, the probability that one useable bias exists gets smaller and smaller.

3.3 Classical attacks exploiting the structure

In this section, we discuss some classical cryptographic attacks that exploit the "structure" of cryptographic primitives. This includes, among others, differential attacks [10], higher-order differential attacks [46], cube attacks [30] and integral attacks [45]. All these attacks have in common that they rely on the fact that an adversary is able to evaluate the cryptographic primitive (equation system) more than once with different inputs. Since Rasta uses different nonce-based matrices and constants for encryption/decryption, these attacks are thwarted.

Differential attacks. Two prerequisites are needed for performing a differential attack. First of all, an attacker has to find a suitable differential characteristic and in the second step, this attacker has to be able to inject the needed differences. For Rasta such a classical differential attack on the permutation is precluded, since each permutation can be at most evaluated once.

Furthermore, even if we consider related-key attacks, where the attacker is able to use different keys under the same nonce and hence is able to evaluate the permutation of Rasta multiple times, it is unlikely that a good exploitable differential characteristic exists following similar arguments as used in Sect. 3.2.

Higher-order differential and cube attacks. Higher-order differential and cube attacks exploit the algebraic degree of the output functions of a cryptographic primitive. Considering the fact that Rasta aims to have a small AND-depth, the output functions of Rasta only have a low degree. For instance, Rasta with depth 4 has an algebraic degree of 16. Having such a low degree would be a major threat for classical permutation based designs, where a single permutation can be evaluated multiple times.

For instance, in a related-key scenario, an attacker could evaluate the same permutation of depth-4 Rasta 2^{16} times to mount a cube attack or exploit the resulting non-random properties in another way. Hence, Rasta is clearly not secure in such a related-key scenario. However, for single-key attacks, each individual output function can only be evaluated once. Therefore, higher-order differential attacks and cube attacks, which rely on evaluating these functions multiple times are effectively hindered.

Integral attacks. Integral attacks exploit the structure of the linear layer in combination with properties of the used S-boxes. In addition to the fact that the used linear layer of Rasta is highly unstructured, it changes for every application of the nonce. Moreover, we only get one evaluation per permutation instance. Both facts make the application of integral attacks on Rasta unlikely.

3.4 On the relative size of block and security parameter for depth 3

In this section we show a guess-and-determine attack to three-round (toy) variants of Rasta where the block size matches the security parameter ($n = k = s$), or the block size slightly exceeds the security parameter. In particular, the attack is applicable to round-reduced variants of Agrasta. This attack is based on a two-round analysis shown in Appendix A and also requires knowledge of only a single keystream block K_n. As an additional prerequisite, it needs a weakness in the matrix M_2 of the affine layer $A_{2,N,j}$. The adversary can query the XOF in an off-line phase to identify a nonce-counter pair (N, i) that yields such a weak matrix. In the following, we briefly describe the details of the attack, including an estimation on the probability of a weak matrix to occur.

We enumerate by X^i the states after the $(i-1)$-th affine layer (e.g., $X^1 = A_{0,N,j}(K)$) and by $Y^i = S(X^i)$ the states after the subsequent S-layer. The attack starts at X^1. We guess $g_1 = \lceil n/2 \rceil$ bits of X^1 at consecutive even bit indices. This allows to formulate all bits of Y^1, i.e., the state after the S layer, as equations linear in the $(n - g_1)$ unknowns from X^1. Through $A_{1,N,j}$, we can derive the full state X^2 also as linear equations in unknowns from X^1.

Table 7. Best found parameters for the guess-and-determine attack on three-round Agrasta. Encs. = encryptions; p_s = Probability of a weak affine layer.

n (bits)	g_1 (bits)	g_2 (bits)	g_3 (bits)	x_r (bits)	y_r (bits)	$\log_2(t)$ (encs.)	$\log_2(p)$
81	41	37	1	4	74	78.19	−18.71
129	65	59	2	6	118	125.76	−55.46
219	110	103	2	6	206	215.39	−66.45
257	129	122	2	6	244	253.58	−66.22
327	164	157	2	6	314	323.86	−65.86

In backward direction, we can formulate K, and from it $X^4 = K \oplus K_N$, and $Y^3 = A^{-1}_{3,N,j}(X^4)$ also as a linear equation system in the unknowns from X^1.

Next, from Y^1, we can guess $g_2 < n/2$ bits of X^2 at consecutive even bit indices. Therefore, we obtain g_2 further equations for the $n - g_1$ remaining unknowns. Moreover, we can formulate $2 \cdot g_2$ bits of Y^2 as linear equations in the unknowns from X^1. We have to choose the $2g_2$ bits in a way such that we obtain a subset of $y_r = g_2 - 2$ consecutive bits of X^3 to depend only on the $2g_2$ bits of Y^2 through $A_{2,N,j}$, the "weak" linear layer.

Let Mat_n denote the set of all non-singular matrices from $\mathbb{F}_2^{n\times n}$. Let $x, y \in \mathbb{F}_2^n$ be Boolean vectors with $x = M \cdot y$. Let $u, v \in \mathbb{F}_2^n$ denote further Boolean vectors that represent masks; we call a bit x_i relevant iff $u_i = 1$ and y_i relevant iff $v_i = 1$. We call a matrix $M \in \mathsf{Mat}_n$ weak iff for all $i, j \in \{0, \ldots, n-1\}$ with $v_i = 1$ and $u_j = 0$, it holds that $M_{i,j} = 0$. So, if M is weak, all relevant bits of x depend on only relevant bits of y.

Naturally, the probability for a linear layer to be weak depends on the numbers of relevant bits x_r in x, relevant bits y_r in y, and the options to distribute them. In M, we need $(n - y_r) \cdot x_r$ zero entries and to have the relevant bits in both Y^2 and X^3 to be located at consecutive positions. Hence, there exist $(n - y_r + 1)$ consecutive bit positions in Y^2 and $(n - x_r + 1)$ positions in X^3. So, we can approximate

$$\mathbb{P}\left[M \twoheadleftarrow \mathsf{Mat}_n : M \text{ is weak}\right] \geq \frac{(n - x_r + 1)(n - y_r + 1)}{2^{(n-y_r)\cdot x_r}}.$$

We search for a region of x_r consecutive relevant bits in X^3 such they depend only on the y_r relevant bits in Y^2. Since those bits are linear in unknowns from X^1, we can formulate also the x_r bits of X^3 as those. Since they are consecutive, we can guess $g_3 = \lfloor (x_r - 2)/2 \rfloor$ bits at consecutive even indices among them. This yields again g_3 equations for the unknowns form X^1. Moreover, the guessed bits allow to formulate $2g_3$ further linear equations for $2g_3$ consecutive bits of Y^3 through the S-layer since we already formulated Y^3 as equations from the backward direction before.

We require that the x_r consecutive bits in X^3 depend only on the $y_r = 2g_2$ bits of Y^2. We guess $g = g_1 + g_2 + g_3 = \lceil n/2 \rceil + g_2 + \lfloor (x_r - 2)/2 \rfloor$ bits at X^1, X^2, and X^3. We obtain $g_2 + g_3 + 2 \cdot g_3$ equations for the $n - \lceil n/2 \rceil$ unknowns of

X^1. The equation system can be solved with computational complexity

$$t = 2^{g_1+g_2+g_3} \cdot (n - \lceil n/2 \rceil)^{2.8} \text{ binary operations.}$$

Rasta with r rounds consists of $(r+1)(n^2+n)$ XORs in the linear layers, $2rn$ XORs in the S-layers, n XORs in the feed-forward as well as $(r+1)n^2$ multiplications by constants in the linear layers, plus rn multiplications in the S-layers. For $r = 3$, this sums to $8n^2 + 14n$ binary operations. We use this figure to compare the effort in terms of three-round Rasta encryptions and optimize the choice of g_2, g_3, and x_r with respect to lowest computational effort while remaining a practical probability that a weak matrix occurs of at least 2^{-80}. The results of the optimal values for the interesting state sizes are provided in Table 7. We implemented a proof of concept of the attack for small state sizes of $n \in \{11, 21\}$, tested it with 100 random keys and weak affine layers each.

3.5 Experiments on small block size variants of Rasta

Number of Monomials. To confirm the analysis made in Section 3.1, we looked at a reduced version with $n = 21$ bits. We start with a state where each bit is a variable and compute one round, that is $S \circ A(x)$. As expected, we found out that all the monomials of degree less or equal to 2 are present in the resulting state. Similarly, we start with a formal state x and compute $S^{-1} \circ A^{-1}(x)$. Since $n = 21$, the degree of S^{-1} is 11. On average on 20 instances (i.e. 20 different linear layers), we obtained that the resulting state contains 1394774 monomials out of the 1401292 monomials of degree less or equal to 11, that is 99.5%.

Practical Gröbner basis experiments. As mentioned above, [11] provided some experiments on computing a Gröbner basis using the algorithm F4 as implemented in Magma. Those experiments, limited to two and three rounds and a block size of at most 101 (for two rounds) and 49 (for three rounds) confirmed our intuition that the security of Rasta is not threatened by this attack vector.

Practical SAT solver experiments. To practically evaluate the performance of automatic solvers for Rasta, we solved small-scale key-recovery challenges with a state-of-the-art SAT solver. Each challenge consists of finding a k-bit key input that generates a randomly generated keystream, for a randomly sampled nonce. Such a challenge is expected to have a valid solution with a probability of roughly 63 %, which was confirmed by our experiments. We translated 8 challenges per depth $d \in \{1, 2, 3\}$ and block size $k \in \{11, 13, \ldots, 41\}$ (if $d = 1$) or $k \in \{11, 13, \ldots, 31\}$ (if $d > 1$) to a 3-SAT instance in conjunctive normal form (CNF) and solved with parallelized SAT solver `plingeling` [9]. The minimum, average, and maximum runtime out of the 8 challenges per parameter configuration is illustrated in Fig. 8. Only challenges that were (a) valid and (b) solved within a time limit of 1 hour are included, so there are no results for $d = 3$ and $k > 25$, and only one result for $d = 2$ and $k > 27$. All runs were evaluated

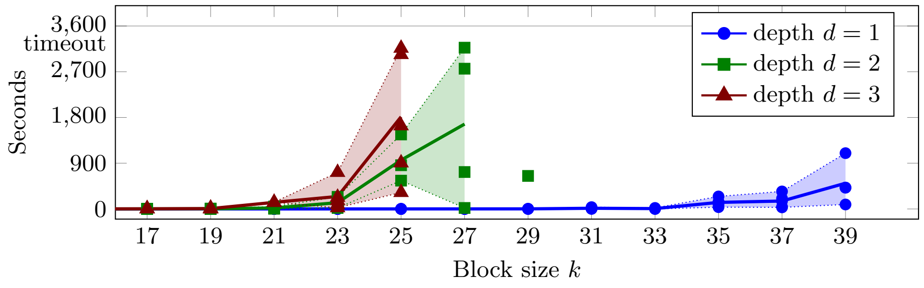

Fig. 8. SAT solver runtime for successful key-recovery on toy parameter sizes: Average, minimum, and maximum runtime for up to 8 challenges per block size.

on Intel Xeon E5-2630v3 CPUs (8 cores max. 3.2 GHz, 92 GB DDR4 RAM). Surprisingly, the solvers performed worse than exhaustive search for all depths $d > 1$. It appears the large, dense linear layers do not interact well with the SAT solver's decision heuristics.

Linear properties. To test the linear properties of the cipher, we consider versions with small block sizes and we compute the linear approximation table of the full cipher. Recall that $LAT[a, b] = (1/2) \times W_{a,b}$ where $W_{a,b}$ is the Walsh coefficient. In our case, we have: $W_{a,b} = \sum_{K \in \mathbb{F}_2^n} (-1)^{a \cdot K \oplus b \cdot K_{N,i}}$, where K is the key and $K_{N,i}$ is the keystream for block i of nonce N.

As explained previously, to measure the resistance of these reduced versions we should not only look at the value of the higher coefficients of their LAT but also see if it is likely that the same linear approximation holds with high probability for various instances. To apprehend these questions, we run the following experiments for variants on 9 and 11 bits and for different number of rounds:

– Pick 50 linear layers at random. For each of them, look for the maximum (in absolute value) of the LAT coefficients. Report the maximum and the minimum over the 50 maximums, together with their average (first three lines of Table 8).
– Pick 50 linear layers at random. Compute the average LAT (that is a LAT where each coefficient is the average of the corresponding (absolute) values of the 50 instances). Look for the maximum of this LAT (line 4 in Table 8).

The obtained results in Table 8 show that the linear properties of variants of 4 and more rounds are close to the one of random permutations.

To visualize this graphically, we use the Pollock representation of the LAT as introduced in [13]. We start by looking at the LAT of specific instances of the small scaled ciphers (Figure 9). On these, we can observe patterns for variants of 1 and 2 rounds, but these disappear for 2 rounds when looking at the average LAT for 50 random instances (see Figure 10). Together with Table 8, this seems to indicate that it is really unlikely that several instances have a bias for the same input and output masks.

Table 8. Experiments on small scaled variants of Rasta. We look at 50 instances of each variant, and for each we note the maximum absolute coefficient of its LAT. We report below the maximum, the minimum and the average of these maximums. We also report the maximum of the average of the LAT.

$n = 9$	2 rounds	3 rounds	4 rounds	5 rounds	random
max	128	64	60	60	60
min	60	50	46	48	50
average	72.72	53.88	53.40	52.96	52.92
max in av. LAT	13.68	13.76	14.00	14.12	14.36
$n = 11$	2 rounds	3 rounds	4 rounds	5 rounds	random
max	384	160	132	130	126
min	184	112	112	110	112
average	256.64	121.92	119.44	117.64	118.40
max in av. LAT	29.44	30.72	29.32	28.96	29.20

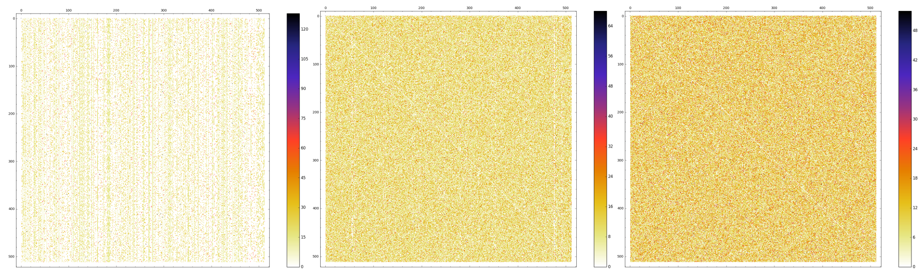
Fig. 9. Pollock representation of the LAT for 1, 2 and 3-round versions of one instance of the cipher on 9 bits.

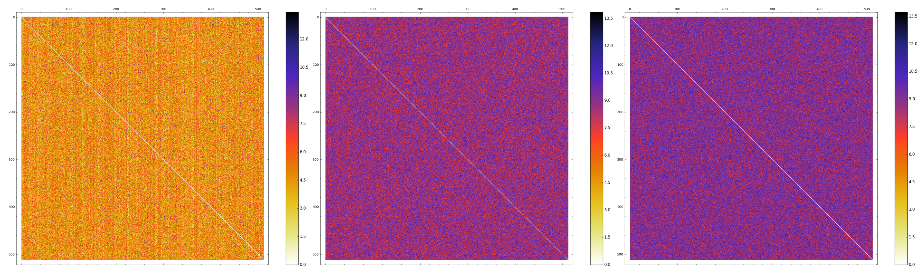
Fig. 10. Pollock representation of the average LAT for 1, 2 and 3-round versions over 50 instances of the cipher on 9 bits.

4 Validation of design approach through benchmarking of FHE use-case

To test the feasibility of our proposed design approach for various choices for the ANDdepth, we implemented Rasta using Helib [40,41], which implements the BGV homomorphic encryption scheme [18] and which was also used to evaluate AES-128 [35,34] and earlier custom designs that minimize the number of multiplications. Our implementation represents each plaintext, ciphertext and key bits as individual HE ciphertexts on which XOR and AND operations are performed. In the HE setting the number of AND gates is not the main determinant of complexity. Instead, the ANDdepth of the circuit largely determines the cost of XOR and AND, where AND is more expensive than XOR. However, due to the high number of XORs in Rasta, the cost of the linear layer is not negligible. In our implementation we use the "method of the four Russians" [1] to reduce the number of HE ciphertext additions from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^2/\log(n))$.

Caveats. All earlier works of custom constructions use Helib for comparative timings. However things are vague in this part of the literature as security levels are not given (automatically determined by Helib from within a wide range of possibilities). Also the number of slots (i.e. blocks processed in parallel) is not under the direct control of the user (and as can be seen from the comparison tables in the literature). Hence we caution the reader to not interpret too much into the detailed timings. These timings should rather be seen as supporting evidence for the practical feasibility of a design approach. In contrast to earlier benchmarks we give the concrete (conjectured) security level of the instantiation of the underlying BGV scheme. We also try to get that BGV security level close to the security level of the cipher. Earlier comparisons were always done with at most 80-bit of security.

Discussion of timing results. A detailed overview of our benchmarks is given in Table 9 and 10. We use the publicly available Helib implementation of LowMC and compare it with our implementation of Rasta in various settings. We try parameters for 80-bit, 128-bit, and 256-bit security. In addition to the case of a pure cipher evaluation (Table 9), we also consider the case where the BGV parameters are chosen such that there is enough "room" for more noise coming from operations that constitute the actual reason (Table 10). For comparability with work in [20,49] we also chose to allow for 7 additional levels for this purpose. Note that we also include the time spent on parts that are not depending on the key, i.e., to generate all the affine layers. For simplicity, we re-use the approach that is used in LowMC to generate random invertible matrices. Only for larger block sizes this is not completely negligible, but it always amounts for less than 10 % of the overall time. This confirms our model to focus on AND-related metrics of those parts of the algorithm that depend on the key.

We cannot directly compare Trivium, Kreyvium and FLIP on our machine as no HElib implementation for them is public. Trivium and Kreyvium numbers

Table 9. Performance comparison of Rasta on Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz CPU with 8GB of RAM. BVG parameters chosen to allow homomorphic evaluation of cipher only. n is the block size or the number of encrypted bits, d is the multiplicative depth, # slots is the number of slots used in HElib, t_{total} is the total running time of the decryption in seconds and includes the time to generate all nonce-dependant computations.Trivium and Kreyvium estimated based on [20], FLIP estimated based on [49], both using LowMCv1 numbers as a point of reference and linear extrapolation.

Cipher	n	d	t_{total}	BGV slots	BGV levels	BGV security
80-bit cipher security						
LowMC v1	128	11	884.7	600	13	64.24
LowMC v2 (low latency)	128	12	546.0	600	14	75.82
LowMC v2 (throughput)	256	12	733.6	600	14	62.83
Trivium	57	12	$\sim$500.0	504	–	–
Trivium	136	13	$\sim$1500.0	682	–	–
FLIP	1	4	$\sim$0.4	378	5	–
Rasta	327	4	110.5	378	5	50.72
Rasta	327	5	206.2	150	7	78.60
Rasta	219	6	159.0	150	7	78.60
Agrasta	81	4	20.0	378	5	50.72
128-bit cipher security						
LowMC v1	256	12	2807.6	720	15	132.26
LowMC v1	256	12	2298.8	720	15	105.72
LowMC v2	256	14	2981.5	720	17	88.01
Kreyvium	46	12	$\sim$1090.0	504	–	–
Kreyvium	125	13	$\sim$2530.0	682	–	–
FLIP	1	4	$\sim$3.3	630	6	–
Rasta	525	5	464.5	330	7	103.97
Rasta	351	6	815.0	600	9	86.37
Agrasta	129	4	50.3	150	5	117.38
256-bit cipher security						
LowMC v2	512	18	8142.7	1285	20	107.67
Kreyvium	Not specified for this security level					
FLIP	Not specified for this security level					
Rasta	703	6	1345.5	720	9	149.43
Agrasta	257	5	1141.1	720	9	212.90

are estimates based on [20], FLIP numbers are estimates based on [49], both using LowMCv1 numbers as a point of reference and linear extrapolation.

As can be seen in the tables, even the very conservative Rasta can in several cases offer significantly lower latency than LowMC or Kreyvium/Trivium. As FLIP is special in this table with the ability to take advantage of producing only a single keystream bit with minimal latency, the latency there is even lower.

Table 10. Performance comparison of Rasta, like Table 9, but BGV parameters are chosen to allow homomorphic evaluation of 7 more levels on top of the cipher evaluation.

Cipher	n	d	t_{total}	BGV slots	BGV levels	BGV security
80-bit cipher security						
LowMC v1	128	11	2011.9	720	20	74.05
LowMC v2 (throughput)	256	12	1721.3	600	21	62.83
Trivium	57	12	~1560.0	504	–	–
Trivium	136	13	~4050.0	682	–	–
FLIP	1	4	~3.5	600	12	–
Rasta	327	4	397.8	224	12	89.57
Rasta	327	4	609.6	600	13	62.83
Rasta	327	5	766.7	600	14	62.83
Rasta	219	6	610.6	600	14	62.83
Agrasta	81	4	98.9	600	12	81.41
128-bit cipher security						
LowMC v1	256	12	3785.2	480	21	106.31
Kreyvium	12	42	~1760.0	504	–	–
Kreyvium	13	124	~4430.0	682	–	–
FLIP	1	4	~39.0	720	13	–
Rasta	525	5	912.1	682	14	90.39
Rasta	351	6	2018.6	720	15	110.74
Agrasta	129	4	217.4	682	12	127.50
256-bit cipher security						
LowMCv2	Too big to run					
Kreyvium	Not specified for this security level					
FLIP	Not specified for this security level					
Rasta	703	6	5543.2	720	16	89.93
Agrasta	257	5	1763.8	1800	15	210.68

However, taking the number of encrypted bits into account the comparison with Rasta seems to be in favour of Rasta.

One interesting thing to note here is that in some cases we were not able to successfully complete the cipher evaluation, despite seemingly moderate parameter sizes. At the 256-bit security level LowMC instances could only be computed purely, but not when 7 additional levels of homomorphic operations are still allowable. With Rasta it was possible. Other designs do not offer parameter-sets for such high security levels.

On the performance in classical settings. Rasta has been designed to minimize the ANDdepth. This is achieved by relying on affine layers, which are newly generated per block that is encrypted. This clearly comes at a cost in classical encryption settings. To get insight on this costs, we have benchmarked our non-optimized implementation that uses SHAKE256 [51] as XOF and the algorithm of Randall [55] for the generation of the matrices on an Intel Core i5-4200U CPU. For Rasta, we measure a performance between 900 000 (6 rounds and 219-bit blocks) and 3 390 000 (6 rounds and 703-bit blocks) cycles per byte. In the case of Agrasta, the performance ranges from 235 000 to 915 000 cycles per byte. Clearly, we expect that the performance can be improved drastically for implementations that just consider fixed block sizes.

5 Conclusion and future work

Summary. We studied Rasta constructions where the substitution layer is chosen to be of low ANDdepth and public and fixed, but each affine layer is different, derived from a public nonce and various counters. Our conclusion is that they are interesting candidates for schemes that try to offer *simultaneously* a very low ANDdepth and a very low number of AND computations per encrypted bit. This contributes to a better understanding of the limits of what concrete symmetric-key constructions can achieve.

Implementations and Applications. Applications for symmetric schemes that minimize metrics like those we consider in this paper are currently investigated in various lines of work [2,4,20,38,49]. To test the applicability of our theoretical work, we chose the FHE setting (using HElib) and benchmarked actual implementations of Rasta, concluding that balanced choices of instantiations appear to be in the same ballpark as other specialized approaches. Our more aggressively parameterized variants termed 'Agrasta' result in our HElib experiments in an improvement of around one order of magnitude. It would be interesting to test applications of Rasta in various other settings, like secure multiparty communication protocols. Due to its low depth it will be especially beneficial for all those protocols where the round-complexity is linear in the ANDdepth of the evaluated circuit (e.g. GMW, tiny-OT).

Cryptanalysis. To better understand the security offered by Rasta, we explored various attack vectors including algebraic attacks, linear approximations and statistical attacks and choose parameters for the instantiations of Rasta to rule them out in a *conservative* way. While we conclude that known attacks do not threaten our design, we encourage further cryptanalysis and also proposed concrete toy versions to that end.

Improving the affine layer. As we have shown, the huge amount of XORs influences performance in targeted applications, and even more so considerably slows down a "plain" implementation of Rasta. New ideas for linear-layer design are needed which impose structure in one way or another which on one hand allows for significantly more efficient implementations while at the same time still resist attacks and allows for arguments against such attacks.

Extending the idea. A natural yet very interesting continuation of this work would be to study the case when the substitution layer is generated from a nonce, or both the substitution layer and the affine layers are derived from a nonce.

Acknowledgments. This research was supported by H2020 project Prismacloud, grant agreement n°644962 and by the Austrian Science Fund (project P26494-N15).

References

1. Albrecht, M.R., Bard, G.V., Hart, W.: Algorithm 898: Efficient multiplication of dense matrices over GF(2). ACM Trans. Math. Softw. 37(1) (2010)
2. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: ASIACRYPT. LNCS, vol. 10031, pp. 191–219 (2016)
3. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: EUROCRYPT. LNCS, vol. 9056, pp. 430–454. Springer (2015)
5. Applebaum, B., Haramaty, N., Ishai, Y., Kushilevitz, E., Vaikuntanathan, V.: Low-complexity cryptographic hash functions. In: ITCS 2017. pp. 7:1–7:31 (2017), `https://doi.org/10.4230/LIPIcs.ITCS.2017.7`
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM J. Comput. 36(4), 845–888 (2006)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (version 3.0) (2011), `http://keccak.noekeon.org`
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak specifications. Submission to NIST (Round 3) (2011), `http://keccak.noekeon.org`
9. Biere, A.: Lingeling, Plingeling and Treengeling entering the SAT Competition 2013. In: Balint, A., Belov, A., Heule, M., Järvisalo, M. (eds.) SAT Competition 2013. vol. B-2013-1, pp. 51–52 (2013), `http://fmv.jku.at/lingeling/`
10. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: CRYPTO. LNCS, vol. 537, pp. 2–21. Springer (1990)
11. Bile, C., Perret, L., Faugère, J.C.: Algebraic cryptanalysis of RASTA. technical report (2017)
12. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In: ASIACRYPT. LNCS, vol. 8873, pp. 63–84. Springer (2014)
13. Biryukov, A., Perrin, L.: On reverse-engineering s-boxes with hidden design criteria or structure. In: CRYPTO. LNCS, vol. 9215, pp. 116–140. Springer (2015)
14. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. J. Cryptology 23(4), 505–518 (2010)
15. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)
16. Boneh, D., Eskandarian, S., Fisch, B.: Post-quantum EPID group signatures from symmetric primitives. Cryptology ePrint Archive, Report 2018/261 (2018), `https://eprint.iacr.org/2018/261`
17. Boyar, J., Peralta, R., Pochuev, D.: On the multiplicative complexity of Boolean functions over the basis (cap, +, 1). Theor. Comput. Sci. 235(1), 43–57 (2000)
18. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. ECCC 18, 111 (2011)

19. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS. pp. 309–325. ACM (2012)
20. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In: FSE. LNCS, vol. 9783, pp. 313–333. Springer (2016)
21. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: CCS. pp. 1825–1842. ACM (2017)
22. Coron, J.S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: PKC. LNCS, vol. 8383, pp. 311–328. Springer (2014)
23. Courtois, N.: Fast algebraic attacks on stream ciphers with linear feedback. In: CRYPTO. LNCS, vol. 2729, pp. 176–194. Springer (2003)
24. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: EUROCRYPT. LNCS, vol. 2656, pp. 345–359. Springer (2003)
25. Daemen, J.: Cipher and hash function design – Strategies based on linear and differential cryptanalysis. Ph.D. thesis, Katholieke Universiteit Leuven (1995)
26. De Cannière, C.: Trivium: A stream cipher construction inspired by block cipher design principles. In: ISC. LNCS, vol. 4176, pp. 171–186. Springer (2006)
27. Derler, D., Ramacher, S., Slamanig, D.: Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In: PQCrypto. pp. 419–440 (2018)
28. Dinur, I., Dunkelman, O., Kranz, T., Leander, G.: Decomposing the ASASA block cipher construction. Cryptology ePrint Archive, Report 2015/507
29. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized interpolation attacks on lowmc. In: ASIACRYPT. LNCS, vol. 9453, pp. 535–560. Springer (2015)
30. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: EUROCRYPT. LNCS, vol. 5479, pp. 278–299. Springer (2009)
31. Dobraunig, C., Eichlseder, M., Mendel, F.: Higher-order cryptanalysis of lowmc. In: ICISC. LNCS, vol. 9558, pp. 87–101. Springer (2015)
32. Duval, S., Lallemand, V., Rotella, Y.: Cryptanalysis of the FLIP family of stream ciphers. In: CRYPTO. LNCS, vol. 9814, pp. 457–475. Springer (2016)
33. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: TCC. LNCS, vol. 3378, pp. 303–324. Springer (2005)
34. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. Cryptology ePrint Archive, Report 2012/099
35. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: CRYPTO. LNCS, vol. 7417, pp. 850–867. Springer (2012)
36. Gilbert, H., Plût, J., Treger, J.: Key-recovery attack on the ASASA cryptosystem with expanding s-boxes. In: CRYPTO. LNCS, vol. 9215, pp. 475–490. Springer (2015)
37. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC. pp. 218–229. ACM (1987)
38. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: Mpc-friendly symmetric key primitives. In: CCS. pp. 430–443. ACM (2016)
39. Grosso, V., Leurent, G., Standaert, F.X., Varici, K.: Ls-designs: Bitslice encryption for efficient masked software implementations. In: FSE. LNCS, vol. 8540, pp. 18–37. Springer (2014)
40. Halevi, S., Shoup, V.: Design and implementation of a homomorphic-encryption library. `https://github.com/shaih/HElib/` (2013)

41. Halevi, S., Shoup, V.: Algorithms in helib. In: CRYPTO. LNCS, vol. 8616, pp. 554–571. Springer (2014)
42. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: TCC. LNCS, vol. 4948, pp. 155–175. Springer (2008)
43. Joux, A., Vitse, V.: A crossbred algorithm for solving boolean polynomial systems. Cryptology ePrint Archive, Report 2017/372
44. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. Cryptology ePrint Archive, Report 2018/475 (2018), `https://eprint.iacr.org/2018/475`
45. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: FSE. LNCS, vol. 2365, pp. 112–127. Springer (2002)
46. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Communications and Cryptography: Two Sides of One Tapestry. pp. 227–233. Kluwer Academic Publishers (1994)
47. Laur, S., Talviste, R., Willemson, J.: From oblivious AES to efficient and secure database join in the multiparty setting. In: ACNS. LNCS, vol. 7954, pp. 84–101. Springer (2013)
48. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT. LNCS, vol. 765, pp. 386–397. Springer (1993)
49. Méaux, P., Journault, A., Standaert, F.X., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: EUROCRYPT. LNCS, vol. 9665, pp. 311–343. Springer (2016)
50. Minaud, B., Derbez, P., Fouque, P.A., Karpman, P.: Key-recovery attacks on ASASA. In: ASIACRYPT. LNCS, vol. 9453, pp. 3–27. Springer (2015)
51. National Institute of Standards and Technology: FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. U.S. Department of Commerce (August 2015)
52. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: CRYPTO. LNCS, vol. 7417, pp. 681–700. Springer (2012)
53. Perrin, L., Promitzer, A., Ramacher, S., Rechberger, C.: Improvements to the linear layer of lowmc: A faster picnic. Cryptology ePrint Archive, Report 2017/1148 (2017), `https://eprint.iacr.org/2017/1148`
54. Raddum, H.: Personal communication (2017)
55. Randall, D.: Efficient generation of random nonsingular matrices. Random Structures & Algorithms 4(1), 111–118 (1993)

A On the relative size of block and security parameter for depth 2

In this section, we give an attack on a 2-round variant of Rasta with a block size equal to the security parameter $n = k = s$. The attack only requires knowledge of one keystream block K_N. Note that it only applies to 2-round variants.

Given the definition of the S-layer, it can be easily seen that making a guess on 1 bit out of 2 of its input (i.e. on first, third, fifth bit and so on) allows to express its output as a linear combination of the $\frac{n}{2}$ unknown bits. In the following, we denote by U, V, X and Y respectively the n-bit state at the input of the first S-layer, its output, the state at the input of the second S-layer and its output. We start by making a guess on $\lceil \frac{n}{2} \rceil$ bits of U, and represent by a variable x_i, $0 \leq i < n - \lceil \frac{n}{2} \rceil$, all the remaining unknown bits. From it, we compute the expression of the key K by inverting the first affine layer. We combine this expression with the knowledge of the keystream block K_N to deduce a linear expression of Y in function of our $n - \lceil \frac{n}{2} \rceil$ variables. Our guess on U allows us to compute an expression of V that is linear in the x_i, from which we deduce an expression of X.

Then, we make a guess on $\lceil \frac{n}{6} \rceil$ consecutive bits[8] of even indexes of X. This guess defines $\lceil \frac{n}{6} \rceil$ linear equations. We then use these guesses to deduce a linear expression of $2 \times \lceil \frac{n}{6} \rceil$ bits of Y that by confronting with the previously obtained expression give $2 \times \lceil \frac{n}{6} \rceil$ linear equations. We then have enough linear equations to be able to solve the system, that we do with $(n - \lceil \frac{n}{2} \rceil)^{2.8}$ binary operations. The total complexity of the attack is then of $2^{\lceil \frac{n}{2} \rceil} \times 2^{\lceil \frac{n}{6} \rceil} \times (n - \lceil \frac{n}{2} \rceil)^{2.8} \simeq 2^{2n/3} (\frac{n}{2})^{2.8}$ binary operations. Applying this attack to a $s = n = k = 41$-bit variant of 2-round Rasta would require $2^{40.1}$ binary operations, which is less than 2^s encryptions. An alternative attack with slightly higher complexity was independently observed by Raddum [54].

B Attacks based on linear approximations – details

In order to get the results presented in Sect. 3.2 - "Bounds for 2 S-Layers (Depth 2)", we exploit the probability that a certain fixed linear characteristic over a randomly generated matrix M_1 exists. Here we show in detail how to compute this probability, which is equal to $(2^n - 1)^{-1}$.

Let $b, c \in \mathbb{F}_2^n \setminus 0$ fixed but arbitrary. In this section, we show how to compute the number of invertible matrix $A \in \mathbb{F}_2^{n \times n}$ such that

$$A \cdot b = c. \tag{6}$$

To begin, we analyze the case b and/or c completely equal to 0. Obviously, if $b = (0, \ldots, 0)$, then the previous equation does not have any solution A if $c \neq (0, \ldots, 0)$, while any matrix A satisfy it if $c = (0, \ldots, 0)$. Similar considerations hold for $c = (0, \ldots, 0)$.

[8] In some cases, $\lfloor \frac{n}{6} \rfloor$ could be sufficient.

In what follows, we denote by b_z for $0 \leq z < n$ the z-bit of b, and by $A_{r,c}$ the bit in row r and in column c of A. Given a vector $x \in \mathbb{F}_2^n$, we denote by $\text{hw}(x)$ its Hamming Weight, that is $\text{hw}(x) = \sum_{i=0}^{n-1} x_i$.

First, some considerations about A. How many different invertible matrices A exist? Denoting by $\hat{m}$ this number, by simple computation we have:

$$\hat{m} = \prod_{i=0}^{n-1} (2^n - 2^i).$$

Indeed, the first row ($i = 0$) of A can assume any value expect all zeros (thus, $2^n - 1$); the second row ($i = 1$) can not be all zeros and it must be different from the first row (thus, $2^n - 2$); the third row ($i = 2$) can not be all zeros and must be different from each (not-null) combinations of the first two rows (thus $2^n - 4$); the i-th row can not be all zeros and must be different from each (not-null) combinations of the first i-1 rows (thus $2^n - 2^i$).

Come back to equation (6). Since $b \neq (0, \ldots, 0)$, there exists (at least one) $0 \leq z < n$ such that $b_z \neq 0$ (that is, $b_z = 1$). Thus, equation (6) can be rewritten as follows for each i with $0 \leq i < n$:

$$A_{i,z} = c_i \oplus \bigoplus_{j \neq z} A_{i,j} \cdot b_j. \tag{7}$$

Fixed $b, c \neq (0, \ldots, 0)$, the expected value of the number m of invertible matrix A is defined as

$$m = \sum_{i=1}^{n} \mathbb{P}[\text{hw}(c) = i] \cdot m_i,$$

where n_i is the number of invertible matrix A that satisfy the previous equality when $\text{hw}(c) = i$ (note that $n_0 = 0$ since $b \neq (0, \ldots, 0)$ for hypothesis). By simple computation, the probability that $\text{hw}(c) = i$ is given by[9]:

$$\mathbb{P}[\text{hw}(c) = i] = \binom{n}{i} \cdot \frac{1}{2^n - 1},$$

where $2^n - 1$ instead of 2^n is justified by the assumption $c \neq 0$.

Consider the case $\text{hw}(c) = 1$, and suppose without loss of generality (w.l.o.g.) that $c_0 = 1$ and $c_i = 0$ for all $i \geq 1$. By equation (7), the first row of A can take 2^{n-1} possible values, since one bit is fixed (note that this row cannot be completely equal to zero, otherwise (7) is not satisfied since $c_0 = 1$). In the same way, the second row of A can take 2^{n-1} possible values. However, since A must be invertible, it cannot be completely equal to zero, that is it can only take $2^{n-1} - 1$ possible values. The third row can take $2^{n-1} - 2$ possible values, since one bit is fixed, it cannot be completely equal to zero or equal to the second row (note that if it is equal to the first row, then it does not solve (7), since $c_0 \neq c_2$).

[9] Remember that $\sum_{i=0}^{n} \binom{n}{i} = 2^n$.

Working in the same way for the next rows of A, we obtain that

$$m_1 = 2^{n-1} \cdot \prod_{i=0}^{n-2} (2^{n-1} - 2^i).$$

Consider now $\mathrm{hw}(c) = 2$, and suppose w.l.o.g. that $c_0 = c_1 = 1$ and $c_i = 0$ for all $i \geq 2$. As before, the first row of A can take 2^{n-1} possible values. The second row of A can take $2^{n-1} - 1$, since it should be different from the previous one. The third row of A can take $2^{n-1} - 2$ possible values, since it cannot be completely equal to zero and it cannot be equal to the sum of the previous two rows (note that this is the only linear combination that solves (7)). The fourth row of A can take $2^{n-1} - 4$ possible values, since it cannot be completely equal to zero and it cannot be equal to the sum of the first two rows, to the sum of the previous three rows, and to the third row. Working in the same way for the next rows of A, we obtain that $m_2 = m_1$.

In the same ways as before, it is possible to prove that m_i is independent of $\mathrm{hw}(c) = i$, that is:

$$m_i = 2^{n-1} \cdot \prod_{j=0}^{n-2} (2^{n-1} - 2^j) \qquad \forall i = 1, \ldots, n.$$

Indeed, let $\mathrm{hw}(c) = i$, and suppose w.l.o.g. that $c_0 = c_1 = \cdots = c_{i-1} = 1$ and $c_j = 0$ for all $j \geq i$. As before, the first row of A can take 2^{n-1} possible values. The second row of A can take $2^{n-1} - 1$, since it should be different from the previous one. Working as before, the $(i-1)$-th row can take only $2^{n-1} - 2^{i-2}$. In a similar way, the i-th row (note that $c_i = 0$ while $c_j = 1$ for $j < i$) can take only $2^{n-1} - 2^{i-1}$, since it cannot be equal to 0 or to any (non-null) linear combination of an even number of the previous row[10]. The $i+1$-th row can take only $2^{n-1} - 2^i$, since it cannot be equal to 0 or to any (non-null) linear combination of an even number of the first $i-1$ rows and of row i. Working in the same way, one obtain the desired result.

Since m_i is independent of $\mathrm{hw}(c)$ (for b and c not completely equal to 0), it follows that

$$m = 2^{n-1} \cdot \prod_{i=0}^{n-2} (2^{n-1} - 2^i) \sum_{j=1}^{n} \binom{n}{j} \cdot \frac{1}{2^n - 1} = 2^{n-1} \cdot \prod_{i=0}^{n-2} (2^{n-1} - 2^i).$$

For the following, note that m can also be rewritten as

$$m = 2^{n-1} \cdot \frac{1}{2^{n-1}} \prod_{i=0}^{n-2} (2^n - 2^{i+1}) = \prod_{i=0}^{n-2} (2^n - 2^{i+1}) = \prod_{i=1}^{n-1} (2^n - 2^i),$$

where we have collected 2^{-1} for each term of the product.

[10] Note that this number is given by $\sum_{j=0}^{\lfloor r/2 \rfloor} \binom{r}{2j} = 2^{r-1}$.

As last thing, fixed $b, c \in \mathbb{F}_2^n \setminus 0$, what is the probability p that an invertible matrix A satisfies (6)? By simple calculation:

$$p = \frac{m}{\hat{m}} = \frac{1}{2^n - 1}.$$

C On the first and the last affine layers A

While the use and importance of the affine layers A between the nonlinear S-layers has been evaluated in the main body of the document, we have a look at the importance of the first and last affine layers A of Rasta in this section. First, we show the importance of the first affine layer by evaluating how this layer influences trivial linearization attacks. Then, we give an attack for a variant of Rasta with depth 2 that omits the first and the last affine layer (essentially a *SAS* construction).

C.1 On the importance of the first affine layer A

If we consider Rasta without the first affine layer, the input to the first S-layer does not change, which could be simply exploited in a trivial linearization attack. Since the output of the first S-layer does not change in this construction, we could simply target these values, plus the secret values of the original key that is added at the end of the permutation. Thus, we now get maximal the following number of different monomials for Rasta with degree ϕ, which is much lower compared to (2):

$$U = k + \sum_{i=0}^{\phi/2} \binom{k}{i}.$$

C.2 On the importance of the first and last affine layers A

One property that comes along with the use of the χ-transformation [25, Section 6.6.3] as S-layer is its low diffusion, namely the fact that one output bit only depends on 3 input bits. As we show in this section, this behaviour can lead to an attack of a 3-step version of the cipher, namely *SAS*. However, it is easy to see that the affine layer is efficiently compensating the low diffusion of the S-layer, making an extension of this attack inefficient.

Consider a known plaintext scenario for which we have access to D keystream blocks $K_{N,0}$ to $K_{N,D-1}$.

We start by making a guess on the g first bits of the secret key K and deduce the value of the first $g-2$ bits of the state at the output of the first S-layer (note here that the value of this state is the same for all the evaluations of the cipher). By combining this guess with the known values of the keystream blocks $K_{N,i}$, we learn the value of the first g bits at the output of the second non-linear layer. Given the properties of the S-layer, we are able to obtain the value of (in average)

$g - 3$ bits[11] of the state entering this operation. At this point, we know $g - 2$ bits at the input of the affine layer, together with $g - 3$ bits of its output. We thus obtain $g - 3$ linear equations with $n - g + 2$ unknowns for each keystream block.

For instance, if we apply this technique to the smaller toy version of Table 4 limited to the *SAS* layers for which $n = 193$, we would have a total of $195 - g$ unknowns and each keystream block would give $g - 3$ equations. The value of D is bounded by $\frac{\sqrt{2^s}}{n}$ where s is the security (here $s = 24$) so to 21 blocks.

If we use Strassen's algorithm to solve the resulting linear system we end up with an attack with a time complexity of:

$$C_T = 2^g \times (193 - g + 2)^{2.8}.$$

The smallest time complexity is obtained by taking $g = 12$, and gives $C_T = 2^{33}$ binary operations and requires $D = 21$ keystream blocks.

It can be easily seen that extending this technique to either an *ASAS* or a *SASA* version of the cipher is made difficult by the changing affine layer.

Consider for instance the *SASA* cipher. In this case, a guess on g consecutive key bits still provides $g - 2$ bits at the input of the first affine layer, but the second affine layer renders impossible to get information on consecutive bits at the end of second S-layer for several keystream blocks, and then to get bits at its input.

[11] This value can be found experimentally by counting the number of bits differing between the possible pre-images of a given g-bit word, for values of g close to 8.