Attacks on Goldreich's Pseudorandom Generators by Grouping and Solving

Ximing Fu[1,2], Mo Li[3], Shihan Lyu[3] and Chuanyi Liu[1,2]

[1] Harbin Institute of Technology, Shenzhen
[2] Pengcheng Laboratory, Shenzhen
{fuximing,liuchuanyi}@hit.edu.cn
[3] The Chinese University of Hong Kong, Shenzhen
{220019160,225010002}@link.cuhk.edu.cn

Abstract. Goldreich's pseudorandom generators (PRGs) with constant locality admit highly parallel implementations, yet the concrete security of instantiations based on the XOR-THR predicate has remained unclear.

In this work, we present novel seed recovery attacks on Goldreich's PRGs instantiated on XOR-THR predicates with $m = n^s$, where m and n are the length of output and input, respectively. By partitioning the output bits into groups according to common input bits, high-biased noisy equations can be derived for the group whose common input bits are all 1s (or all 0s). Leveraging two solvers tailored to these equations, we achieve the seed recovery attacks, which needs roughly $2^{n(1-\log_2(1+\frac{2s}{\sqrt{2\pi(b-s)}}))}$ calls to Gaussian elimination when the input length of the THR predicate $b \geq (n-s)^{1/s} + s - 1$ with small stretch s.

Applying our attack to the asymptotic XOR-MAJ family proposed in STOC 2016 yields complexity $2^{n\left(1-\log_2\left(1+\sqrt{\frac{s}{18\pi}}\right)\right)} \leq 2^{0.82n}$, i.e., at least a $2^{0.18n}$ speedup over exhaustive search for any stretch $s > 1$. We also deploy our attack on an instance used in the construction of silent oblivious transfer protocols (Eurocrypt 2024) with $n = 256$. This attack is capable of breaking the instance using approximately $2^{31.3}$ calls to Gaussian elimination over 244 variables. We successfully implemented the attack on a cluster of 14 CPU cores, recovering the seed in 71 hours, demonstrating the attack's efficiency.

Beyond PRGs, with appropriate adaptations our method extends to FiLIP stream ciphers instantiated with THR-related predicates. Our evaluation indicates that most FiLIP instances do not meet their claimed security. For instance, the configuration XOR_{100}-$\mathsf{THR}_{11,22}$-$\mathsf{THR}_{11,22}$ is broken in about 2^{59} calls to Gaussian elimination over 357 variables despite a 397-bit key and a claimed 80-bit security level.

Keywords: Goldreich's PRGs · XOR-THR predicates · noisy equations · FiLIP cipher

1 Introduction

Pseudorandom Generators (PRGs) are a cornerstone of cryptography, taking as input a short random seed of length n and producing $m \gg n$ pseudorandom output bits. Goldreich [40] introduced a family of one-way functions with the *locality* property, where each output bit is determined by applying a fixed $d(n)$-ary predicate to a randomly selected subset of $d(n)$ input bits. PRGs constructed from such predicates, known as Goldreich's PRGs, inherit this locality. In particular, when $d(n)$ is constant, Goldreich's PRGs can be realized with constant overhead, enabling efficient parallelization. This feature makes them highly scalable across multiple computing cores and has drawn considerable attention.

The development of Goldreich's PRGs has been closely tied to the exploration of different predicates. Cryan and Miltersen [29] first considered the existence of constant locality Goldreich's PRGs and obtained a negative result: any PRGs with $d = 3$ cannot resist statistical linear tests for $m \geq 4n$. Mossel et al. [58] further ruled out the case $d = 4$ for $m \geq 24n$, but also proposed a candidate instantiation with $d = 5$, based on the XOR-AND predicate $P_5(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$, achieving superlinear stretch.

A natural and ambitious goal is to further achieve *polynomial stretch*, i.e., $m = n^s$ for factor $s > 1$. Goldreich's PRGs with polynomial stretch have garnered special attention due to their wide cryptographic applications, such as secure computation with constant computational overhead [46,5,7], indistinguishability obfuscation (iO) [12,13,61,62,52], MPC-friendly primitives [57,41,2], and cryptographic capsules [19]. Applebaum [3] first proved the existence of weak PRGs with polynomial stretch $m = n^s$ for some specific s, having a distinguishing gap of at most $1/n^s$. This result was later strengthened for $s < 1.25$ using non-degenerate predicates [4], and further extended to $s < 1.5$ for the special case of P_5 [59]. Subsequently, the general polynomial regime $m = n^s$ was considered, and the XOR-THR predicate was proposed as the candidate underlying predicate with constant locality [8], where the challenge of $\mathsf{XOR}_a\text{-}\mathsf{MAJ}_b$ with $a \geq 5s$ and $b > 36s$ was proposed to achieve s-pseudorandomness with proved resistance to existent attacks. Due to the high efficiency, XOR-MAJ was applied to specific encryption schemes such as Oblivious Transfer (OT) construction [21] and symmetric ciphers [56,39].

Since the inception of Goldreich's PRGs [40], cryptanalysis has been an integral part of the research on predicate design and PRG construction. Early works were focused on asymptotic security. However, concrete security is also crucial, especially for parameter selection in explicit encryption schemes. For instance, although both 2^n and 1.1^n are exponential in n, the latter only provides 18-bit security even when n reaches 128. For PRGs based on the P_5 predicate, Guess-and-Determine (GAD) attack [28] and the improved version [66], Guess-and-Decode (GADec), have been proposed, with complexities $O(2^{\frac{1}{2}n^{2-s}})$ and $O(2^{\frac{73}{288}n^{2-s}})$, respectively. For the XOR-MAJ predicate, the GAD attack succeeds with complexity $O(n^\omega 2^{n\frac{b-s}{b-1}})$ [28]. However, as noted in [21], these attacks are

specifically designed for predicates with very small locality (ranging from 5 to 8), and their complexity grows significantly for predicates with larger locality. So, it remains an open problem whether efficient attacks exist for XOR-MAJ predicates with large locality.

1.1 Related Work for Cryptanalysis

Generally, recovering the seed of a Goldreich's pseudorandom generator can be reduced to finding a solution for a Constraint Satisfaction Problem (CSP). It has been proven NP-hard to exactly solve CSPs [23,50] or approximate the solutions of CSPs [9,10,11]. The recovery of Goldreich's pseudorandom generators can also be reduced to solving the multivariate quadratic (MQ) problem, especially for those instantiated on P_5 predicate. Solving the MQ problem is NP-hard [38,37] and has been extensively studied, including algebraic attacks [35,34], Crossbred algorithm [47,64,63] and polynomial methods [51,31,30].

Regarding asymptotical attacks on Goldreich's PRGs, linear test has been a popular cryptanalysis technique. The bias is used to detect the imbalance of the output of the PRGs. Cryan and Miltersen showed that the generator with $d = 3$ and $m \geq 4n + 1$ can be broken by the linear test [29]. Algebraic attacks [25,27,26] are another technique and have been widely studied and applied to the cryptanalysis of stream ciphers. The algebraic attack was formulated in a general notion in [8]. As a countermeasure, XOR-MAJ was proposed as the candidate underlying predicate of Goldreich's PRGs to obtain s-pseudorandomness for $m = n^s$.

Couteau et al. first considered the concrete security and proposed the Guess-and-Determine (GAD) attack on Goldreich's PRGs, mainly focused on those instantiated on the P_5 predicate [28]. They observed that by guessing one bit in the quadratic term of the P_5 predicate, one linear equation over partial seed bits can be derived. When a sufficiently large number of seed bits are guessed, a collection of linear equations can be obtained and solved for recovering the seed. This attack was extended by GADec attack [66] in two ways. Firstly, through a more thorough analysis, the number of guesses can be reduced. In addition, the decoding method for low density parity check (LDPC) codes was exploited to speed up the solving phase. The quadratic equations were exploited in a clever way to avoid the computation of a-priori information for iterative decoding. These two attacks achieve significant speedup over brute force on P_5 based PRGs. However, the efficiency can be damaged significantly when applying to Goldreich's PRGs instantiated on the XOR-MAJ predicate. Both the GAD and GADec attacks aim to solve noise-free equations, relying on guessing partial bits. This is easily done for P_5 predicate as 1-bit guessing in the quadratic term can make the equation linear while it needs much more guesses for the XOR-MAJ predicate. In addition, the reliance on low density of GADec makes it hard to execute on XOR-MAJ predicate. Hence, as justified in [66,21,1], GADec can only work for very small locality.

1.2 Related Work for Solving Noisy Equations

In this paper, our attack approach fundamentally differs from GAD and GADec in that it builds on solving noisy linear equations. This task can be reduced to the Learning Parity with Noise (LPN) problem, which is well known to be hard [20] and has been extensively studied.

The widely used algorithms for LPN are the BKW method [17] and its improved variants [36,49,53,42,67,43], which are generally regarded as an effective family of solvers in practice. However, BKW-like algorithms require a number of equations that vastly exceeds the number of variables. This requirement does not align with the PRG case considered in this paper (as demonstrated in several attack instances), where the output is relatively short for security guarantees. In addition, they require memory costs as large as time costs, further making them difficult to implement due to the physical memory limit. As discussed in [33], it seems infeasible to perform any algorithm with memory consumption larger than 2^{60} bits in practice.

The other type of LPN solvers is called Gauss-like algorithms [22,33]. Gauss method [22] applies Gaussian elimination on a randomly selected system of equations and verifies each solution by testing the correct probability. As analyzed in [33], the Gauss method still requires the same equation cost as its time cost, which motivated the proposal of Pooled-Gauss [33] to reduce the number of required equations. In this paper, we adopt the modified Gauss and Pooled-Gauss methods for solving noisy equations, where the verification is performed by substituting solutions into the PRG schemes rather than testing the correct probability of noisy equations in [22,33]. In addition, we use fewer equations than in [33] with detailed analysis.

Another alternative for solving LPN problems is Information Set Decoding (ISD) [15,16,54,55,44], which was originally developed for the syndrome decoding problem and was equipped into LPN solvers [33]. In the literature, two types of ISD solvers are considered: half-distance decoding and full decoding. The half-distance variant is generally more efficient; for example, in MMT [54] the complexity is $2^{0.05364c}$ for half-distance decoding compared to $2^{0.1116c}$ for full decoding, where c denotes the code length. Hence, we only consider the half distance decoding for comparison and omit the recent improvements over full decoding such as Sieving [55,44,32], which strengthens our results. In this paper, we adopt the half distance decoding versions of ISD solvers including MMT [54], BJMM [15] and Well-Pooled MMT [33] for comparison.

1.3 Our Contributions

In this work, we propose a novel seed recovery attack on Goldreich's PRGs, namely Group-and-Solve (GAS) attack, specifically targeting those instantiated with the $\mathsf{XOR\text{-}THR}$ predicates. The core of our approach is a simple yet highly efficient grouping technique, which significantly amplifies the bias in the grouped noisy equations. Specifically, we demonstrate that when in a group the common r bits into the THR predicate are 1s, the correlation between the output and the

corresponding linear expression, $L(x) = 1$, increases substantially in this group. With this amplified bias, we enhance the Gauss and SubGauss algorithms to solve the resulting equations more efficiently, enabling successful seed recovery, costing $2^{n(1-\log_2(1+\frac{2s}{\sqrt{2\pi(b-s)}}))}$ calls to Gaussian elimination when $b \geq (n-s)^{1/s} + s - 1$ with small stretch s, where b denotes the input length of the THR predicate.

We apply our attack to the asymptotic XOR-MAJ family proposed in STOC 2016 [8] and to an OT primitive from Eurocrypt 2024 [21]. For this family, our results demonstrate that certain PRGs with the XOR-MAJ predicate can be attacked with complexity upper bounded by $2^{0.82n}$, significantly speeding up over brute force. For the instance at Eurocrypt 2024, by selecting appropriate parameters, we show that a PRG with a 256-bit secret seed can be practically recovered with approximately $2^{31.3}$ calls to Gaussian elimination. We implement this attack on a single PC with 14 CPU cores and the seed was recovered in 71 hours.

Finally, we present key recovery attacks on FiLIP ciphers [39], specifically targeting instances based on THR-related predicates. By partial guess, the input taken into the THR predicate together with the corresponding output can be grouped similar to that in the PRG case, such that the similar attack can be applied. For most of these instances, our attacks are efficient and successful. For example, the FiLIP instance of key size 1987 bits, can be broken with 2^{95} calls to Gaussian elimination. Similarly, FiLIP with the XOR-THR-THR predicate, with a key size of 913, can be broken with 2^{98} calls to Gaussian elimination. In both cases, the assumed security level is $\lambda = 128$.

The rest of the paper is organized as follows. Section 2 introduces the necessary preliminaries, and Section 3 presents our GAS attack. In Section 4, we apply the attack to the XOR-MAJ family proposed in STOC 2016 and to an instance used for silent OTs (Eurocrypt 2024), together with implementation details. Section 5 extends our approach to key-recovery attacks on the FiLIP cipher instantiated with XOR-THR and XOR-THR-THR predicates. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we fix the notation and recall basic definitions that will be used throughout the paper, with a focus on concepts relevant to Goldreich's pseudorandom generators (PRGs) and seed recovery attacks.

All computations in the paper are carried out over the finite field $\mathbb{F}_2$, with addition denoted by $+$ (or $\sum$). For a vector $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$ and an index tuple $\sigma = (i_1, \ldots, i_k)$ with $i_j \in \{1, \ldots, n\}$, we write

$$x[\sigma] \triangleq (x_{i_1}, \ldots, x_{i_k}),$$

where all indices are assumed to be distinct unless otherwise stated. In the special case of a consecutive range $\sigma = (a, a+1, \ldots, b)$ with $1 \leq a \leq b \leq n$, we use the shorthand

$$x[a{:}b] \triangleq (x_a, x_{a+1}, \ldots, x_b).$$

When unambiguous, we write $x = 1$ (resp. $x = 0$) to indicate that all entries of the vector x equal 1 (resp. 0).

2.1 Predicates and Goldreich's PRGs

A PRG maps a seed $x \in \{0,1\}^n$ to an output $y \in \{0,1\}^m$, with $m \gg n$. Goldreich's PRG with constant locality d is defined by a predicate $P : \{0,1\}^d \mapsto \{0,1\}$ and index tuples $\sigma_1, \ldots, \sigma_m$, where each $\sigma_i = (i_1, \ldots, i_d)$ specifies d distinct input indices. The i-th output bit is then given by

$$y_i \;=\; P\big(x[\sigma_i]\big).$$

For positive integers d and $t \le d$, and for $x \in \{0,1\}^d$, define

$$\mathsf{XOR}_d(x) \triangleq \sum x, \qquad \mathsf{THR}_{t,d}(x) \triangleq \begin{cases} 1, & \text{if } w_H(x) \ge t, \\ 0, & \text{otherwise,} \end{cases}$$

where $w_H(x)$ is the Hamming weight of x. For odd d, we simply write $\mathsf{MAJ}_d \triangleq \mathsf{THR}_{\frac{d+1}{2},d}$. The XOR-THR predicate can be viewed as a straightforward composition of the basic predicates above. Specifically, for positive integers a, b and $t \le b$, and for $x \in \{0,1\}^{a+b}$,

$$\mathsf{XOR}_a\text{-}\mathsf{THR}_{t,b}(x) \triangleq \mathsf{XOR}_a\big(x[1{:}a]\big) \;+\; \mathsf{THR}_{t,b}\big(x[a+1{:}a+b]\big).$$

Analogous definitions apply to XOR_a-MAJ_b (for odd b) and to other composed predicates.

Since in this work we mainly consider composed predicates (e.g., XOR_a-MAJ_b), it is convenient to write

$$\sigma_i = (\sigma_{i,1}, \sigma_{i,2}),$$

with $|\sigma_{i,1}| = a$ and $|\sigma_{i,2}| = b$.

2.2 Seed Recovery Attack and Noisy Equation

The seed-recovery problem for Goldreich's PRGs is defined as follows: given a predicate P with locality d, seed length n, index vectors $\Sigma = (\sigma_1, \ldots, \sigma_m)$, and output string $y = (y_1, \ldots, y_m)$, the task is to recover the seed x. In this work we restrict attention to the polynomial-stretch regime, where m grows polynomially in n (e.g., $m = n^s$ for some stretch factor $s \ge 1$).

Our attack is to recast the seed recovery problem as solving noisy linear equations. For each $i \in \{1, \ldots, m\}$, the PRG process produces an output bit $y_i = P(x[\sigma_i])$, which can be interpreted as a noisy linear equation by linearizing the nonlinear part of the predicate P. In the case of XOR-THR predicates, since the XOR part is already linear, the linearization applies only to the THR component.

More concretely, for a linear function $L : \{0,1\}^b \mapsto \{0,1\}$, the i-th process can be written as the noisy equation

$$y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + L(x[\sigma_{i,2}]),$$

with holding probability

$$p = \Pr[\mathsf{THR}_{t,b}(x[\sigma_{i,2}]) = L(x[\sigma_{i,2}])].$$

In line with standard terminology, the correlation between THR and function L is $\mathrm{cor} = 2p - 1$ and the bias is $\epsilon = p - \frac{1}{2}$.

Note that the correlation (and hence the bias) may be negative. In such cases, considering $L(x) + 1$ simply flips the sign, so without loss of generality, we restrict attention to the nonnegative case.

In this work, we primarily adopt the special case $L(x) = 1$, partly because it simplifies the subsequent analysis, and partly because it is already optimal for all concrete instances we consider in this paper. A detailed discussion can be found in Section 3.

2.3 Complexity Estimates

It's quite nontrivial to provide a complexity estimate that satisfies everyone, especially in the concrete analysis. In this paper, we provide several approaches for complexity evaluation, aiming to comprehensively cover various scenarios.

Similar to guess-and-determine (GAD) [28] and guess-and-decode (GADec) [66] methods, our attack can be reduced to a sequence of calls to Gaussian elimination (GE). Hence, we use the number of GEs, termed as *GE complexity*, to estimate the complexity and omit the discussion of GE, following the approach in GAD. This method can simplify the asymptotic analysis and comparison with related works, making readers focused on the improvement, especially the comparison with GAD and GADec. So in the theoretic analysis, we adopt this estimate metric.

Of course, the estimate using number of GEs has limitations, as GEs of different sizes incur significantly different costs, both asymptotically and concretely. Another method is to count the bit operations, termed as *bit complexity*, making the complexity multiplied with a factor n^ω (e.g., $\omega = 3$) where n is the matrix size over which GE is applies. However, as discussed in [43], this estimate does not match the practical costs of modern microprocessors, which are usually measured by clock cycles, since word-oriented and SIMD instructions can perform many bit operations in a single instruction. In practice, a row addition (involving many XOR operations) in Gaussian elimination requires only one or a few cycles.

In terms of bit complexity, we also need to consider the bit costs of encryptions, as we often argue an attack is successful when it outperforms brute force. Hence, the ratio of the bit complexity over the bit costs of an encryption, i.e., the equivalent number of encryptions, is also usually measured as the complexity. We call this metric the *encryption complexity*.

For the attacks on FiLIP ciphers, we provide GE complexity, bit complexity and encryption complexity for the sake of completeness.

However, we believe that the number of GEs more clearly reflects the concrete costs, at least this is the case in the instances we have attacked in this paper.

3 Group-and-Solve Attack on Goldreich's PRGs with XOR-THR Predicate

In this section, we introduce *Group-and-Solve* (GAS), a seed recovery attack on Goldreich's PRGs instantiated with XOR-THR predicates. We first present the key observation motivating the attack, and then present the GAS framework along with its analysis.

3.1 Key Observation: High-biased Noisy Equations after Grouping

It is well known that the MAJ_b predicate is balanced when b is odd, outputting 0 and 1 with equal probability under uniform input. Consequently, its correlation with constant functions is zero, and thus exhibits no bias. However, we observe that when the truth table is grouped according to some common input indices, certain groups may become significantly imbalanced, which in turn yields noisy equations with high bias. To illustrate this observation, consider the following example.

Example 1. Consider the predicate $P = \mathsf{MAJ}_3$ with input chosen from $\{x_1, x_2, x_3, x_4\}$, among which two bits are 1s and the other two bits are 0s. Without loss of generality, we assume that $x_1 = x_2 = 1$ and $x_3 = x_4 = 0$. We show the grouping technique in Fig. 1.

From the truth table in Fig. 1a, the predicate MAJ_3 outputs 1 and 0 with equal probability over a randomly chosen subset $\sigma \subset \{1, 2, 3, 4\}$ of size 3, consistent with the fact that MAJ is balanced.

When we focus on the sub truth table that shares the common input bit $x_1 = 1$, as shown in Fig. 1b, the output equals 1 with probability 2/3, i.e.,

$$\Pr[P(x[\sigma_i]) = 1 \mid x_1 = 1] = \frac{\binom{2-1}{1}\binom{4-2}{2-1}}{\binom{4-1}{3-1}} = \frac{2}{3}.$$

Thus, the corresponding bias is $\epsilon = \Pr[P(x[\sigma_i]) = 1 \mid x_1 = 1] - \frac{1}{2} = \frac{1}{6}$. Analogous results hold for the sub truth tables that share any other common input bit.

When we further group the outputs according to two common input bits, we obtain the sub truth tables shown in Fig. 1c. In particular, the table corresponding to common inputs x_1 and x_2 always outputs 1, while the one for x_3 and x_4 always outputs 0; the remaining tables are balanced.

Example 1 demonstrates that grouping by common input bits can turn a balanced Boolean function into an imbalanced one. Moreover, if the number of

input	output
$x_1x_2x_3$	1
$x_1x_2x_4$	1
$x_1x_3x_4$	0
$x_2x_3x_4$	0

(a) The full truth table.

common x_1

input	output
$x_1x_2x_3$	1
$x_1x_2x_4$	1
$x_1x_3x_4$	0

common x_2

input	output
$x_1x_2x_3$	1
$x_1x_2x_4$	1
$x_2x_3x_4$	0

common x_3

input	output
$x_1x_2x_3$	1
$x_1x_3x_4$	0
$x_2x_3x_4$	0

common x_4

input	output
$x_1x_2x_4$	1
$x_1x_3x_4$	0
$x_2x_3x_4$	0

(b) Grouping based on a single common input.

common x_1x_2

input	output
$x_1x_2x_3$	1
$x_1x_2x_4$	1

common x_1x_3

input	output
$x_1x_2x_3$	1
$x_1x_3x_4$	0

common x_1x_4

input	output
$x_1x_2x_4$	1
$x_1x_3x_4$	0

common x_2x_3

input	output
$x_1x_2x_3$	1
$x_2x_3x_4$	0

common x_2x_4

input	output
$x_1x_2x_4$	1
$x_2x_3x_4$	0

common x_3x_4

input	output
$x_1x_3x_4$	0
$x_2x_3x_4$	0

(c) Grouping based on two common inputs.

Fig. 1: An example of the grouping technique. The order of the inputs is omitted, as MAJ is a symmetric predicate.

common bits is sufficiently large (for instance, $(b+1)/2$), a group may become deterministic: the nonlinear component outputs all 1's (or all 0's) on that group, and the corresponding noisy equations are in fact error-free. The Guess-and-Determine (GAD) attack of [28] exploits such error-free linear equations to recover seeds, and the Guess-and-Decode (GADec) method of [66] extends this idea to quadratic equations. However, for XOR-THR predicates, especially when b is large, the guessing cost of these approaches becomes significant.

We further observe that the bias grows very quickly with the number of common bits: even a modest number of common bits set to 1 suffices to produce highly biased noisy equations, which can then be attacked using LPN-solving techniques to recover the seed. To formalize this phenomenon, we relate the number of common bits to the holding probability for general $\mathsf{XOR}_a\text{-}\mathsf{THR}_{t,b}$ predicates in Proposition 1. In what follows we assume, without loss of generality, that the common bits are all 1; the case where they are all 0 is analogous.

Proposition 1. *For any seed x of Hamming weight w, b bits are randomly taken as input to the* THR *part of the PRG. In the group whose common input $x[\rho_i] = 1$, where $\rho_i \subseteq \sigma_{i,2}$ and $|\rho_i| = r$, the following noisy equation $y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + 1$ holds with probability*

$$\frac{\sum_{i=t-r}^{\min(b-r,w-r)} \binom{w-r}{i}\binom{n-w}{b-r-i}}{\binom{n-r}{b-r}}, \tag{1}$$

where the probability is over the random chosen subset whose Hamming weight is $w-r$.

Proof. As there are total w 1s in the seed, of which r bits are in the THR predicate, the THR predicate outputs 1 if and only if at least $t-r$ out of the remaining $w-r$ bits of 1s appear in the THR predicate. Then we enumerate all the possible number out of $w-r$ 1s, i.e., i from $t-r$ to $\min(b-r, w-r)$, where the $b-r$ term denote the input-length constraint of the THR predicate while the $w-r$ term denotes the Hamming weight constraint. The other $b-r-i$ positions are taken from the $n-w$ 0s. Hence, Pproposition 1 is proved.

3.2 Group-and-Solve Attack

Motivated by the observation above, we now present the details of our GAS attack. The overall framework, illustrated in Figure 2, consists of two cascaded phases: an initial *grouping* phase followed by a *solving* phase.

Both phases are conceptually simple. In the grouping phase, noisy equations are grouped according to their common input indices. Then, in the solving phase, each group forms an LPN instance, which is subsequently solved using efficient algorithms.

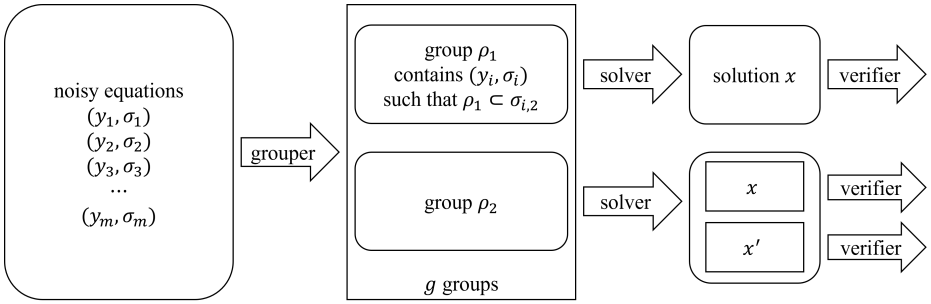


Fig. 2: Framework of Group-and-Solve Attack

Grouping Phase The process of the grouping phase is summarized in Algorithm 1. It takes as input two parameters: the number of common indices r and the number of groups g. Then, g index sets $(\rho_1, \rho_2, \ldots, \rho_g)$ are randomly chosen, each of size r, and the corresponding groups are denoted by $(\mathcal{G}_1, \ldots, \mathcal{G}_g)$. Finally, for each $i \in \{1, \ldots, m\}$ and each j, the noisy equation

$$y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + 1$$

Algorithm 1 Equation Grouping

Input: Number of fixing bits r, number of groups g

for $j = 1$ **to** g **do**
  $\rho_j \leftarrow$ randomly chosen a set of r distinct indices, each $\in [1, n]$
  $\mathcal{G}_j \leftarrow \emptyset$
  for $i = 1$ **to** m **do**
    if $\rho_j \subseteq \sigma_{i,2}$ **then**
      $\mathcal{G}_j$.append$(y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + 1)$
    end if
  end for
end for
return $(\mathcal{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_g)$ and $(\rho_1, \rho_2 \cdots \rho_g)$

is assigned to group $\mathcal{G}_j$ whenever $\rho_j \subseteq \sigma_{i,2}$.

The time complexity of Algorithm 1 is

$$T_{\text{group}} = mg. \tag{2}$$

On average, each noisy equation is assigned to $\binom{b}{r}$ out of the total $\binom{n}{r}$ groups. Hence, the expected number of equations in each group is

$$e = m \cdot \frac{\binom{b}{r}}{\binom{n}{r}} = \frac{n^s \binom{b}{r}}{\binom{n}{r}}. \tag{3}$$

Solving Phase After the grouping phase, some groups may exhibit high correlation with $L(x) = 1$. Equivalently, the LPN instances defined by these groups have higher bias and are therefore easier to solve. However, since we do not know which groups have this property, the solving phase must attempt to solve each LPN instance. Remarkably, with an appropriate strategy for selecting these attempts, one can expect to find the correct solution with low overall complexity.

In terms of the choice of LPN solvers, we consider three classes: BKW, Gauss, and ISD. Traditional solvers such as BKW [17] are not well suited for the grouped equations, as the number of equations in each group may be insufficient. To address this, we introduce two solvers, *Gauss* and *SubGauss*, which are slightly modified from [33] to handle high-bias, small-size noisy equations. Since these solvers are straightforward, we present them as part of the complete attack process.

The ISD solver can also be employed in this setting, and we compare it with our approach in Section 4.

System Solving: We introduce a generic routine, `System_Solving`, which takes as input a noisy equation pool $\mathcal{G}$, the number of variables k, and a constraint parameter c. The routine randomly selects $\min(k, c)$ noisy equations from $\mathcal{G}$ to form a subsystem, treats it as noise-free, and applies Gaussian elimination to obtain candidate solutions, resolving any remaining variables by

Algorithm 2 Attack with Gauss

Input: Number of fixed bits r

$e \leftarrow \frac{n^s\binom{b}{r}}{\binom{n}{r}}$,$p \leftarrow \frac{1}{2^{b-r}}\sum_{i=t-r}^{b-r}\binom{b-r}{i}$,$g \leftarrow \frac{2^r}{p^{\min\{e,n-r\}}}$
Grouping Phase:
$(\mathcal{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_g), (\rho_1, \rho_2 \cdots \rho_g) \leftarrow$ `Equation Grouping` (r, g)
Solving Phase:
for $j = 1$ **to** g **do**
  $X_j \leftarrow$ `System Solving` $(\mathcal{G}_j, n-r, e)$
  for $x \in X_j$ **do**
    if $P(x[\sigma_i]|x[\rho_i] = 1) = y_i$ for all $i \in [1, m]$ **then**
      return x
    end if
  end for
end for
return $\perp$

enumeration. This routine serves as a core subprocedure repeatedly invoked within our attack framework, and we omit explicit pseudocode because the procedure itself is elementary.

Gauss Solver: The Gauss solver invokes `System_Solving` once for each output group produced by Algorithm 1. The resulting complete GAS algorithm with the Gauss solver is summarized in Algorithm 2, where r is a tunable parameter whose selection is discussed in Section 3.3.

At the beginning of the attack, we compute the probability p using Eq. (1) and the expected number of equations per group, e, using Eq. (3). Then, we set

$$g = \frac{\binom{n}{r}}{\binom{w}{r}p^{\min\{n-r,e\}}}. \tag{4}$$

The grouping phase uses (r, g) as inputs to Algorithm 1, resulting in g groups, each containing approximately e equations, computed by Eq. (3). When $e > n-r$, then randomly chosen $e-r$ equations are used. For each group $\mathcal{G}_j$, $\min(n-r, e)$ equations are used to form a noisy linear system. In the solving phase, we perform `System_Solving` with the parameters $n-r$ and e on the instance to obtain candidate seeds, denoted as X_j. Each candidate seed $\hat{x} \in X_j$ is verified by checking whether $P(\hat{x}[\sigma_i]|\hat{x}[\rho_i] = 1) = y_i$ holds for all $i \in \{1, 2, \ldots, m\}$. Note that when $e < n-r$, $n-r-e$ variables need to be guessed.

SubGauss Solver: In Gauss solver, each Gaussian elimination is performed on a fresh group of noisy equations, consuming large number of groups and hence incurring high grouping complexity. Then we introduce the SubGauss solver. SubGauss is also based on Gaussian elimination, but uses fewer groups with each containing more equations. Concretely, the required number of groups is

$$g = \frac{E}{E-1}\frac{\binom{n}{r}}{\binom{w}{r}} \tag{5}$$

Algorithm 3 Attack with SubGauss

Input: Number of fixing bits r

$e \leftarrow \frac{n^s\binom{b}{r}}{\binom{n}{r}}$,$p \leftarrow \frac{1}{2^{b-r}}\sum_{i=t-r}^{b-r}\binom{b-r}{i}$,$g \leftarrow \frac{E}{E-1}\frac{\binom{n}{r}}{\binom{w}{r}}$
Grouping Phase:
$(\mathcal{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_g), (\rho_1, \rho_2 \cdots \rho_g) \leftarrow$ `Equation Grouping` (r, g)
Solving Phase
for $j = 1$ **to** g **do**
  for $i = 1$ **to** $\frac{1}{p^{\min\{ep, n-r\}}}$ **do**
    $X_j \leftarrow$ `System Solving` $(\mathcal{G}_j, n-r, ep)$
    for $x \in X_j$ **do**
      if $P(x[\sigma_i]|x[\rho_i] = 1) = y_i$ for all $i \in [1, m]$ **then**
        return x
      end if
    end for
  end for
end for
return $\perp$

and the required number of equations in each group is $(n-r)/p$, where E denotes the base of natural logarithm. Compute p by Eq. (1) and e by Eq. (3).

As shown in Algorithm 3, in the *grouping phase*, `Equation Grouping` (Algorithm 1) is performed, so that g groups, each containing e equations, are obtained. If $e > (n-r)/p$, then randomly chosen $(n-r)/p$ equations in each group are used.

In the *solving phase*, similar to the Gauss Solver case, perform `System_Solving` with the parameters $n-r$ and ep on randomly chosen $\min(n-r, ep)$ equations, which is executed $1/p^{ep}$ times for each guess and each group. Each candidate seed $\hat{x} \in X_j$ is verified by checking whether $P(\hat{x}[\sigma_i]|\hat{x}[\rho_i] = 1) = y_i$ holds for all $i \in \{1, 2, \ldots, m\}$. Note that when $e < (n-r)/p$, $n-r-ep$ variables need to be guessed.

3.3 Analysis of Complexity and Success Rate

In this section, we analyze the complexity and success rate of the GAS attack with both Gauss and SubGauss solvers. We further discuss how to choose appropriate parameters and solvers for different instances.

GAS with Gauss Solver The total complexity of our GAS attack is clearly given by the sum of the complexities of the two phases. Revisiting Algorithm 2, the complexity of the solving phase is

$$\begin{cases} g \cdot (T_g + T_v), & e \geq n-r, \\ g \cdot 2^{n-r-e}(T_g + T_v), & e < n-r, \end{cases}$$

where T_g and T_v denote the cost of Gaussian elimination and verification, respectively. Verifying a solution incurs small cost, i.e., roughly $a + b < n$ bit operations for each substitution. Hence, here we only consider the cost of Gaussian elimination, which is no more than $(\min(e, n-r))^\omega$, where ω is a constant. The complexity of the grouping phase is $n^s g$. Substituting g from Eq. (4), the total complexity is

$$T_{Gauss} = \begin{cases} \frac{\binom{n}{r}}{\binom{w}{r} p^{n-r}} (n^s + (n-r)^\omega), & e \geq n-r, \\ \frac{\binom{n}{r}}{\binom{w}{r} p^{e}} (n^s + 2^{n-r-e} e^\omega), & e < n-r. \end{cases} \tag{6}$$

An attack succeeds when a group simultaneously satisfies the following two conditions: (1) the r-bit setting of the group is correct (i.e., the r common bits are all ones); (2) all noisy equations selected from the group hold. Each group satisfies condition (1) with probability $\frac{\binom{w}{r}}{\binom{n}{r}}$, while condition (2) holds with probability $p^{\min(e,n-r)}$. So each group satisfies the above two conditions simultaneously with the probability $\frac{\binom{w}{r}}{\binom{n}{r}} p^{\min(e,n-r)}$, which from Eq. (4) is $1/g$. And the success rate of the attack with Gauss solver is

$$p_{sucGauss} = 1 - \left(1 - \frac{1}{g}\right)^g \approx 1 - \frac{1}{E},$$

where E denotes the base of natural logarithm, and the approximation holds for sufficiently large g.

Increasing r increases p while decreasing e, according to Eq. (1) and Eq. (3), respectively. Accordingly, the Gauss-solver attack reduces to choosing an appropriate r that minimizes T_{Gauss} (Eq. (6)) while achieving the target success rate $\frac{E-1}{E}$.

We remark that the success rate can be improved by imposing more groups and hence larger time complexity. More concretely, if zg groups are used, the success rate can reach up to $1 - (1 - \frac{1}{g})^{zg} \approx 1 - \frac{1}{E^z}$, while the time complexity increases by a factor of z.

GAS with SubGauss Solver Revisiting Algorithm 3, the complexity of the solving phase is

$$\begin{cases} g \cdot \frac{1}{p^{n-r}} (n-r)^\omega, & e \geq (n-r)/p, \\ g \cdot 2^{n-r-ep} \frac{1}{p^{ep}} (ep)^\omega, & e < (n-r)/p. \end{cases}$$

And, as the complexity of *grouping phase* is $n^s g$, by substituting $g = \frac{E}{E-1} \frac{\binom{n}{r}}{\binom{w}{r}}$ (Eq. (5)), the total complexity of attack with SubGauss solver is

$$T_{SubGauss} = \begin{cases} \frac{E}{E-1} \frac{\binom{n}{r}}{\binom{w}{r}} (n^s + \frac{1}{p^{n-r}} (n-r)^\omega), & e \geq (n-r)/p, \\ \frac{E}{E-1} \frac{\binom{n}{r}}{\binom{w}{r}} (n^s + \frac{2^{n-r-ep}}{p^{ep}} (ep)^\omega), & e < (n-r)/p. \end{cases} \tag{7}$$

Now we analyze the success rate of the GAS attack with SubGauss solver. The attack succeeds when: (1) the r-bit setting is correct (r bits are ones) and (2) in the correct r-bit setting, at least one choice of ep equations are correct. As is discussed in the case of Gauss solver, condition (1) holds with probability $\frac{\binom{w}{r}}{\binom{n}{r}}$. Regarding condition (2), let $e' = \min(e, (n-r)/p)$, each out of $1/p^{e'p}$ trials succeeds with the probability

$$\begin{aligned}\frac{\sum_{i=e'p}^{e'}\binom{e'}{i}p^i(1-p)^{e'-i}\binom{i}{e'p}}{\binom{e'}{e'p}} &= \sum_{i=e'p}^{e'}\binom{e'-e'p}{i-e'p}p^i(1-p)^{e'-i} \\ &= p^{e'p}\sum_{i=0}^{e'-e'p}\binom{e'-e'p}{i}p^i(1-p)^{e'-k-i} \\ &= p^{e'p}.\end{aligned}$$

Hence, the latter case holds with the probability $1-\left(1-p^{e'p}\right)^{1/p^{e'p}} = 1-\frac{1}{E}$. As a consequence, each group satisfies the two conditions simultaneously with probability $\frac{\binom{w}{r}}{\binom{n}{r}}(1-\frac{1}{E}) = \frac{1}{g}$. Summed up, the success rate of the attack with SubGauss solver is

$$p_{sucSubGauss} = 1-\left(1-\frac{1}{g}\right)^g \approx 1-\frac{1}{E},$$

where E denotes the base of natural logarithm.

Similarly, the attack success rate with SubGauss can be improved by imposing more groups. The success rate can reach $1-\frac{1}{E^z}$ if zg groups are used and the time complexity is magnified by a factor of z.

Note that the above analysis is derived under the independence hypothesis, i.e., each choice of $e'p$ out of e' equations is independent. The independence hypothesis only holds when the equation number in each group is sufficiently large. Observe that the SubGauss solver is adapted from the Pooled Gauss technique in [33], whose analysis relies on the same assumption. As discussed in [33], dependencies do not noticeably affect performance and success probability. Our experimental results on the Eurocrypt 2024 instance further support this observation.

Discussion on Parameters: r and w As the input of the attack algorithm, r is a crucial parameter for the attack complexity. In the attack on instances with concrete parameters, r is chosen to minimize the complexities in Eq. (6) and Eq. (7), respectively. This optimization problem can be easily solved by progressively testing different values of r starting from 1 to $\min(w,t)$ with $O(n)$ complexity, which is negligible.

In the attack, we explicitly use the Hamming weight w of the seed, which is actually unknown to the attacker. For $w \geq n/2$, the complexities in Eq. (6)

and Eq. (7) decrease as w increases, since the holding probability p increases by Proposition 1. For $w < n/2$, we can turn to consider the correlation of THR predicate to 0 rather than to 1 and the solving complexity decreases as w decreases. Hence, we only consider $w = n/2$ as it is the worst case. In practice, as w is unknown, the attacks with correlation to 1 and 0 can be executed in parallel. Once the seed is recovered, all programs can end. As a randomly chosen seed is balanced with probability $\frac{\binom{n}{n/2}}{2^n} \approx \frac{1}{\sqrt{2\pi n}}$, this attack benefits from the imbalance of the seed in most cases.

Choice of Solvers The two solvers, Gauss and SubGauss, perform well in different scenarios, which are discussed as follows.

When the stretch s is small and hence the output is short, e is small even for small r, the solving phase dominates the cost of the attack. Revisiting the bit-fixing correlation attacks instantiated with Gauss and SubGauss solves, given the same parameter r and hence the same e,

$$\frac{T_{Gauss}}{T_{SubGauss}} = \frac{E-1}{E}\frac{1}{(2p)^{e-ep}}.$$

As $p > 1/2$, T_{Gauss} over $T_{SubGauss}$ is usually smaller than 1, hence the Gauss solver outperforms the SubGauss solver.

However, when the stretch s is large and hence the output is sufficiently long, e may be as large as $n-r$ for a larger r. The attack instantiated with SubGauss solver is obviously more efficient than that of the Gauss solver as they share the same solving complexity (with a $\frac{E}{E-1}$ factor) but the grouping complexity of SubGauss is reduced by a factor of $\frac{E}{E-1}\frac{1}{p^{n-r}}$.

The above results are confirmed by the attacks on the instances in Section 4 and Section 5.

3.4 Asymptotic Analysis

This section examines the asymptotic complexity of our attack when the stretch factor s is small. In this regime, each group contains only a limited number of noisy equations, so we focus exclusively on the Gauss solver.

We first consider the worst-case setting $w = n/2$, as discussed in Section 3.3.

Theorem 1. *Let r^* be the maximal r with the constraint $(b-r+1)^r \geq (n-s)^{r-s+1}$, for sufficiently large n, omitting the n^s and n^ω terms, the complexity is*

$$T_{\mathsf{xt}} \approx \frac{2^n}{\left(1+\frac{2r^*}{\sqrt{2\pi(b-r^*)}}\right)^{n-r^*}} \approx 2^{n(1-\log_2(1+\frac{2r^*}{\sqrt{2\pi(b-r^*)}}))}. \tag{8}$$

Theorem 1 can be proved by checking that the expected number of equations exceeds n and by applying Stirling's approximation with the full proof as follows.

Proof. Let $r = s-1$, $(b-r+1)^r \geq (n-s)^{r-s+1} = 1$. Hence, $r^* \geq s-1$. For the case $r^* = s-1$, the expected equations in each group is $e = n^s \frac{\binom{b}{s-1}}{\binom{n}{s-1}} > n > n-r^*$. For the case $r^* \geq s$, the expected equation number in each group is

$$\begin{aligned} e = n^s \frac{\binom{b}{r^*}}{\binom{n}{r^*}} &= \frac{n^s \cdot b \cdots (b-r^*+1)}{n \cdot (n-1) \cdots (n-s+1) \cdot (n-s) \cdots (n-r^*+1)} \\ &> \frac{(b-r^*+1)^{r^*}}{(n-s)^{r^*-s}} \geq n-s > n-r^*. \end{aligned}$$

In summary, the expected number of equations in each group is larger than $n-r^*$ for any valid r^*.

As s is small (hence r^* is also small), with $w = n/2$, $\frac{\binom{n}{r^*}}{\binom{w}{r^*}} \approx 2^{r^*}$, substituted into Equ. (6), the total complexity is $T_{xt} = \frac{2^n}{(2p)^{n-r^*}}$, achieving a $(2p+1)^{n-r^*}$ speed up over brute force. Hence, Theorem 1 can be reduced to $p = \frac{\sum_{i=t-r^*}^{b-r^*} \binom{w-r^*}{i}\binom{n-w}{b-r^*-i}}{\binom{n-r^*}{b-r^*}} \approx \frac{1}{2} + \frac{r^*}{\sqrt{2\pi(b-r^*)}}$, proved as follows.

Substituting $r = r^*$ and $w = n/2$ into Equ. (1), we have

$$p = \frac{\sum_{i=t-r}^{b-r} \binom{w-r}{i}\binom{n-w}{b-r-i}}{\binom{n-r}{b-r}} = \frac{\sum_{i=t-r^*}^{b-r^*} \binom{n/2-r^*}{i}\binom{n/2}{b-r^*-i}}{\binom{n-r^*}{b-r^*}}.$$

Note that

$$\begin{aligned} \frac{\binom{n/2-r^*}{i}\binom{n/2}{b-r^*-i}}{\binom{n-r^*}{b-r^*}} &= \frac{\frac{(n/2-r^*)!(n/2)!}{i!(n/2-r^*-i)!(b-r^*-i)!(n/2-(b-r^*-i))!}}{\frac{(n-r^*)!}{(b-r^*)!(n-b)!}} \\ &= \frac{\frac{(n-b)!}{(n/2-r^*-i)!(n/2-(b-r^*-i))!}\frac{(b-r^*)!}{i!(b-r^*-i)!}}{\frac{(n-r^*)!}{(n/2-r^*)!(n/2)!}} \\ &= \frac{\binom{n-b}{n/2-r^*-i}\binom{b-r^*}{i}}{\binom{n-r^*}{n/2-r^*}}. \end{aligned}$$

By Stirling's approximation,

$$\binom{n-r^*}{n/2-r^*} = \frac{(n-r^*)!}{(n/2-r^*)!(n/2)!} \approx \frac{2^{n-r^*+1}}{\sqrt{2\pi(n-r^*)}}$$

and

$$\binom{n-b}{n/2-r^*-i} = \frac{(n-b)!}{(n/2-r^*-i)!(n/2-(b-r^*-i))!} \approx \frac{2^{n-b+1}}{\sqrt{2\pi(n-b)}},$$

We have

$$\frac{\binom{n/2-r^*}{i}\binom{n/2}{b-r^*-i}}{\binom{n-r^*}{b-r^*}} \approx \frac{\binom{b-r^*}{i}}{2^{b-r^*}},$$

where the approximation $\sqrt{\frac{n-b}{n-r^*}} \approx 1$ is used, for sufficiently large n.

Then we have

$$p \approx \frac{1}{2^{b-r^*}} \sum_{i=t-r^*}^{b-r^*} \binom{b-r^*}{i}.$$

As $\binom{b-r^*}{i} = \binom{b-r^*}{b-r^*-i}$,

$$p \approx \frac{1}{2^{b-r^*}} \sum_{i=t-r^*}^{b-r^*} \binom{b-r^*}{b-r^*-i} = \frac{1}{2^{b-r^*}} \sum_{i=0}^{b-t} \binom{b-r^*}{i}.$$

As a consequence,

$$\begin{aligned} 2p &\approx \frac{1}{2^{b-r^*}} \sum_{i=0}^{b-t} \binom{b-r^*}{i} + \frac{1}{2^{b-r^*}} \sum_{i=t-r^*}^{b-r^*} \binom{b-r^*}{i} \\ &= 1 + \frac{1}{2^{b-r^*}} \sum_{i=t-r^*}^{b-t} \binom{b-r^*}{i}. \end{aligned}$$

Substituting $t = \frac{b+1}{2}$, for each $i \in [t-r^*, b-t] - [\frac{b+1}{2} - r^*, \frac{b-1}{2}]$, by Stirling's approximation, $\binom{b-r^*}{i} \approx \frac{2^{b-r^*+1}}{\sqrt{2\pi(b-r^*)}}$.

Hence, $2p \approx 1 + \frac{2s}{\sqrt{2\pi(b-r^*)}}$, i.e, $p = \frac{1}{2} + \frac{s}{\sqrt{2\pi(b-r^*)}}$. Substituting p into Equ. (6), we have the complexity

$$\begin{aligned} T_{xt} &= \frac{2^n}{\left(1 + \frac{2r^*}{\sqrt{2\pi(b-r^*)}}\right)^{n-r^*}} \\ &= 2^{n(1-\log_2(1+\frac{2r^*}{\sqrt{2\pi(b-r^*)}}))} \left(1 + \frac{2r^*}{\sqrt{2\pi(b-r^*)}}\right)^{r^*}, \end{aligned}$$

When s is constant or small (relatively to b), omitting the $\left(1 + \frac{2r^*}{\sqrt{2\pi(b-r^*)}}\right)^{r^*}$ term, the complexity is

$$T_{\mathsf{xt}} \approx 2^{n(1-\log_2(1+\frac{2r^*}{\sqrt{2\pi(b-r^*)}}))}.$$

To obtain further asymptotic results, we focus on the special case where $(n-s)^{1/s} + s \leq b \leq n/2$. Here we assume that s is an integer; otherwise, we take $s = \lfloor s \rfloor$.

Corollary 1. *For a sufficiently large n, small (relatively to b) $s > 1$ and $b \geq (n-s)^{1/s} + s - 1$, omitting the n^s and n^ω terms, the complexity is*

$$T_{\mathsf{xt}} \approx \frac{2^n}{\left(1 + \frac{2s}{\sqrt{2\pi(b-s)}}\right)^{n-s}} \approx 2^{n(1-\log_2(1+\frac{2s}{\sqrt{2\pi(b-s)}}))}. \tag{9}$$

Proof. By choosing $r = s$, the expected equations in each group is

$$\begin{aligned} e &= n^s \frac{\binom{b}{s}}{\binom{n}{s}} \\ &= n^s \frac{b \cdot (b-1) \cdots (b-s+1)}{n \cdot (n-1) \cdots (n-s+1)} \\ &> \frac{n^s \cdot (b-s+1)^s}{n^s} \\ &= (b-s+1)^s \\ &\geq n-s. \end{aligned}$$

Corollary 1 can be proved by replacing r^* with s from Theorem 1.

Roughly speaking, if $n = O(b^s)$, the attack is sub-exponential. This may appear counterintuitive at first glance as it may be more secure for larger b. It seems that the condition $b \geq (n-s)^{1/s} + s - 1$ is quite strict, as in Goldreich's PRGs b is expected to be constant. We stress this case due to the fact that this setting covers a wide range of parameters in concrete constructions. Take the instance parameter in [21] (as shown in Section 4.2) as example, where $(n, b, t, s) = (256, 64, 32, 5)$. Immediately, $b = 64 > n^{1/s} + s - 1 = 7$. It can be verified that all the parameters for FiLIP cipher (shown in Table 5) satisfy $n^{1/s} + s \leq b \leq n/2$.

It can be verified from Theorem 1 that for $s = 1$ ($r^* = 0$), the complexity is 2^n, which is the trivial upper bound of complexity. In addition, the constraint $b \geq n^{1/s} + s - 1$ does not hold for $s = 1$. Here, we provide a tight asymptotic analysis for small b and small stretch s.

Theorem 2. *For $s < 2$ and sufficiently large n with $b < (n-s)^{1/s} + s - 1$, omitting the n^s and n^ω terms, the complexity is upper bounded by*

$$T_{\mathsf{xt}} \approx \frac{2^n}{\left(1 + \frac{2s}{\sqrt{2\pi(b-s)}}\right)^{(b-s+1)^s}}. \tag{10}$$

Proof. Let $r = s$, then the number of equations in each group is $e = n^s \frac{\binom{b}{s}}{\binom{n}{s}} > (b-s+1)^s$. The remaining proof follows that for Theorem 1.

Theorem 2 tells that when $n > (b-s+1)^s + s$, our attack can not achieve exponential speed up with n over brute force. Extremely, when b and s are constant, the attack can only provide constant speed up over brute force. Here, we suggest parameters with the constraint $n > (b-s+1)^s + s$ to limit the speedup achieved by GAS..

In fact, for constant locality d and output length $m = n^s$ with fixed $s > 1$, the shrinking-set attack introduced by Applebaum, Ishai, and Kushilevitz [6] and further analyzed by Zichron [68] yields a generic $2^{o(n)}$-time distinguisher.

The self-correction method introduced by Bogdanov and Qiao [18] also yields a $2^{o(n)}$-time inversion attack. Our comparison with brute force does not establish an asymptotic improvement over these generic attacks. This section characterizes the asymptotic complexity and parameter dependence of GAS. The next section will demonstrate the main strength of GAS: its concrete efficiency.

4 Attacks on XOR-THR Based Schemes

In this section, we apply our attack to specific instances of Goldreich's PRGs, including the XOR-MAJ family introduced at STOC 2016 and a concrete instance used in the construction of silent Oblivious Transfers (OTs) at Eurocrypt 2024.

4.1 Attack on the XOR-MAJ Family Proposed at STOC 2016

At STOC 2016 the authors proposed an asymptotic family of XOR_a-MAJ_b predicates [8], claiming security for parameters satisfying $a \geq 5s$ and $b > 36s$. In what follows we use the minimal parameter choice $a = 5s$, $b = 37s$ as a running example, and report results for several stretch factors s and key sizes n. For each parameter setting we choose r to minimize the overall complexity; the corresponding holding probability p and the expected number of equations per group e are computed from Eq. (1) and (3), respectively.

Note that the number of equations generated by grouping may exceed the amount actually required for solving. For example, when $(n, a, b) = (80, 8, 57)$, the Gauss solver needs $e = n - r = 74$ equations, while grouping yields 86. In this case, as mentioned in Section 3.2, we only use 74/86 of the total $m = n^s = 80^{1.5} \approx 716$ output bits, reducing the complexity of the grouping phase by the same factor. Similar savings occur whenever the number of available equations exceeds the requirement. The numerical results are summarized in Table 1.

The results first illustrate the effect of grouping with a concrete instance. Consider $(n, a, b) = (65, 8, 57)$, for which the majority threshold is 29. If we group by fixing a set of 17 common input indices that are equal to 1, the holding probability of the resulting noisy linear equations can reach 0.8010. In this setting an inversion can be carried out with roughly 2^{43} calls to Gaussian elimination, demonstrating the practical efficiency of our GAS attack. The results further validate our analysis about solver choice in Section 3.3. When the stretch is small (e.g., $s = 1$), the Gauss solver outperforms the SubGauss solver, whereas for longer outputs (e.g., $s = 2$) the SubGauss solver becomes superior.

To further promote the concrete cryptanalysis of Goldreich's PRGs, and building on both our analysis and that of [8], we propose a set of challenge parameters, listed in Table 2.

Asymptotic Analysis The asymptotic complexity of our attack on this family can be obtained directly by applying Theorem 1. For the sake of simplifying the analysis, we consider $b = 37s$. Let $r^* = \max(r|(37s - r + 1)^r \geq (n - s)^{r-s+1})$, the asymptotic complexity is $2^{n(1-\log_2(1+\frac{2r^*}{\sqrt{2\pi(b-r^*)}}))}$.

Table 1: The complexities of attacks on the XOR-MAJ family with $a = 5s$ and $b = 37s$ for different stretches s. In order to guarantee valid Majority predicate, b is set to the smallest odd integer for $b \geq 37s$. The output length is $m = n^s$. p denotes the probability each equations holds. e denotes the expected number of equation in each group and ep is the expected number of correct equations. T_{eg} denotes the complexity of grouping phase and T_s denotes the number of calls to Gaussian elimination and substitution for verification in the solving phase. For $n = 65$, we use $w = 32$ as for $w = 33$ we can turn to use the noisy equations correlated to 0s, as discussed in Sec. 3.3. Similarly, we use $w = 42$ for $n = 85$.

s	(n, a, b)	Solver	r	p	e	ep	T_{eg}	T_s
1	(42,5,37)	Gauss	5	0.6325	21	13	2^{25}	$\mathbf{2^{35}}$
		SubGauss	6	0.6625	18	12	2^{13}	$\mathbf{2^{38}}$
	(80,5,37)	Gauss	1	0.5488	37	20	2^{39}	$\mathbf{2^{75}}$
		SubGauss	1	0.5488	37	20	2^{8}	$\mathbf{2^{78}}$
	(128,5,37)	Gauss	1	0.5559	37	20	2^{39}	$\mathbf{2^{122}}$
		SubGauss	1	0.5559	37	20	2^{9}	$\mathbf{2^{126}}$
1.25	(54,7,47)	Gauss	9	0.7247	37	27	2^{35}	$\mathbf{2^{35}}$
		SubGauss	6	0.6438	60	39	2^{14}	$\mathbf{2^{41}}$
	(80,7,47)	Gauss	3	0.6165	47	29	2^{44}	$\mathbf{2^{66}}$
		SubGauss	2	0.5769	81	47	2^{11}	$\mathbf{2^{71}}$
	(128,7,47)	Gauss	2	0.5943	57	33	2^{54}	$\mathbf{2^{114}}$
		SubGauss	1	0.5468	157	86	2^{10}	$\mathbf{2^{118}}$
1.5	(65*,8,57)	Gauss	17	0.8010	39	31	2^{43}	$\mathbf{2^{43}}$
		SubGauss	15	0.7423	55	41	2^{28}	$\mathbf{2^{46}}$
	(80,8,57)	Gauss	6	0.6813	86	58	2^{57}	$\mathbf{2^{47}}$
		SubGauss	5	0.6503	124	80	2^{15}	$\mathbf{2^{52}}$
	(128,8,57)	Gauss	3	0.6207	124	77	$\mathbf{2^{99}}$	2^{89}
		SubGauss	2	0.5801	284	164	2^{13}	$\mathbf{2^{102}}$
2	(85*,10,75)	Gauss	25	0.8972	174	156	$\mathbf{2^{55}}$	2^{42}
		SubGauss	24	0.8783	208	182	$\mathbf{2^{45}}$	2^{43}
	(128,10,75)	Gauss	9	0.7708	107	83	2^{64}	$\mathbf{2^{62}}$
		SubGauss	8	0.7425	193	143	2^{23}	$\mathbf{2^{61}}$
	(256,10,75)	Gauss	4	0.6561	455	298	$\mathbf{2^{173}}$	2^{157}
		SubGauss	4	0.6561	455	298	2^{21}	$\mathbf{2^{158}}$

Similar to the analysis in Section 3.4, when $37s \geq (n-s)^{1/s} + s - 1$, i.e., $n \leq (36s-1)^s + s$, the asymptotic complexity can be naturally computed by Eq. (8) as

$$T_{cha} = \frac{2^n}{\left(1 + \sqrt{\frac{s}{18\pi}}\right)^{n-s}} \approx 2^{n(1-\log_2(1+\sqrt{\frac{s}{18\pi}}))}. \tag{11}$$

As $s \geq 1$, we obtain the complexity upper bound

$$T_{upper} \approx 2^{0.82n}.$$

In other words, our attack can achieve at least $2^{0.18n}$ speedup over brute force under this case.

Table 2: Challenge parameters for concrete cryptanalysis. We adopt $a = 5s$ and minimal odd $b \geq 37s$ [8].

λ (Security Level)	n (Variable Number)	a (Linear Part)	b (Nonlinear Part)	s (Stretch)
80	85	5	37	1
	91	7	47	1.25
	105	8	57	1.5
	146	10	75	2
128	133	5	37	1
	141	7	47	1.25
	155	8	57	1.5
	205	10	75	2
256	263	5	37	1
	285	7	47	1.25
	303	8	57	1.5
	365	10	75	2

Figure 3 presents representative results that validate our analysis. The asymptotic complexity closely matches the concrete values, particularly for small factor, e.g., $s = 1$ and $s = 2$. There is a gap between the concrete complexity and the asymptotic analysis because b (=111) is close to n while the asymptotic analysis is accurate when $b \ll n$. The similar result occurs when n is small for $s = 2$. The results also tell that our attack is sensitive to s, i.e., the attack complexity drops significantly as s increases.

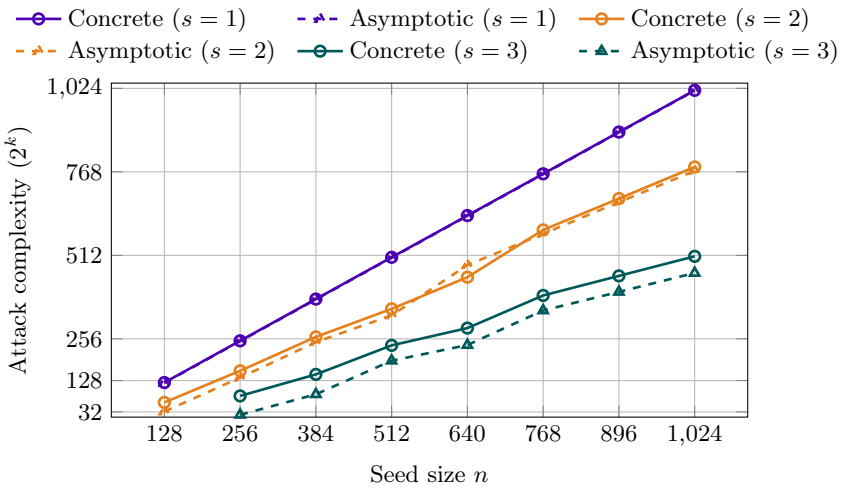


Fig. 3: Complexities by concrete computation and asymptotic analysis.

Comparison with GAD We compare our method with the GAD method [28] and list the results in Figure 4. The comparisons show that our attack substantially outperforms GAD for $s = 2$ and $s = 3$; for example, the complexity of GAD at $s = 3$ already exceeds that of our method at $s = 2$. We do not include a comparison with GADec [66], since it is only applicable when b is very small (e.g., $b = 3, 4$), as stated in [66,21,1], whereas instances here use predicates with much larger locality.

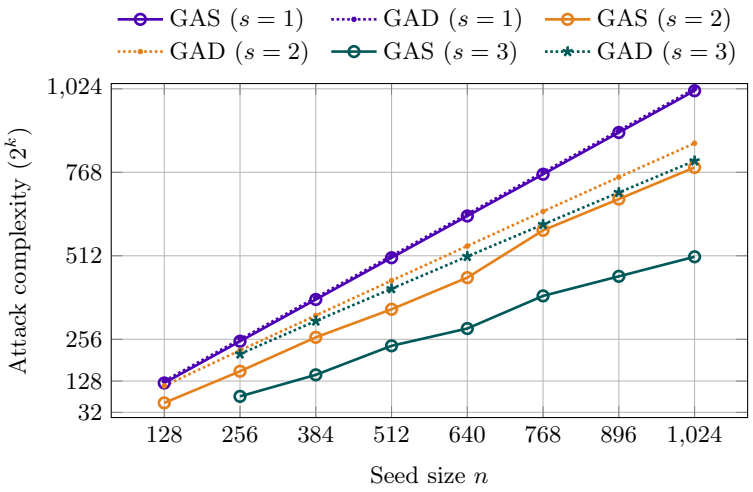


Fig. 4: Comparison of our attack with GAD.

4.2 Attack on the Instance at Eurocrypt 2024

The XOR_a-$\mathsf{THR}_{t,b}$ predicate with the parameter $(n, m, b, t) = (256, 2^{40}, 64, 32)$ was used for constructing fast silent OT [21].

Before the implementation, we first need to determine the parameter r. For each choice of r, the corresponding values of p and e can be computed using Eq. (1) and Eq. (3), respectively. The complexities for Gauss and SubGauss are derived from Eq. (6) and Eq. (7), respectively. For clarity, we present the complexities of the grouping phase and solving phase in separate columns. The results are summarized in Table 3.

From Table 3, we know that for $r \geq 8$, the holding probability $P \geq 0.8629$, i.e., cor > 0.5, the noisy equation we obtain is optimal, according to the Parseval's Theorem [14][4].

[4] The Parseval's Theorem tells that the sum of squared correlations of a Boolean function with all possible linear functions (including the constant function) is 1. Hence, the linear approximation whose square correlation exceeds 0.5 is optimal. The corresponding probability threshold is $\frac{1+\sqrt{0.5}}{2} \approx 0.85$.

Table 3: The parameters for solving the instances [21] and comparison with other methods. D denotes the number of used output bits. e is the expected number of equations in each group and ep denotes the expected number of correct equations in each group. T_{eg} denotes the cost of grouping phase. T_s denotes the number of calls to Gaussian elimination used in the solving phase. For the ISD solvers, e denotes the code length.

Solver	r	p	D	e	ep	T_{eg}	T_s
Gauss	12	0.9495	$2^{33.1}$	244	232	$2^{63.8}$	$2^{30.6}$
	13	0.9629	$2^{35.4}$	243	234	$2^{62.1}$	$2^{26.7}$
	14	0.9736	$2^{37.6}$	242	236	$2^{61.5}$	$2^{23.9}$
SubGauss	12	0.9495	$2^{33.2}$	257	244	$2^{46.3}$	$2^{31.3}$
	13	0.9629	$2^{35.4}$	252	243	$2^{49.5}$	$2^{27.4}$
	14	0.9736	$2^{37.6}$	249	242	$2^{52.9}$	$2^{24.6}$
GAD [28]	∅	∅	∅	∅	∅	∅	2^{131}
MMT* [54]	12	0.9495	$2^{33.6}$	343	326	$2^{46.7}$	$2^{31.5}$
	13	0.9629	$2^{35.7}$	315	303	$2^{49.2}$	$2^{30.4}$
	14	0.9736	$2^{37.9}$	293	285	$2^{52.4}$	$2^{30.3}$
BJMM⋆ [15]	12	0.9495	$2^{33.6}$	343	326	$2^{46.7}$	$2^{30.0}$
	13	0.9629	$2^{35.7}$	315	303	$2^{49.2}$	$2^{29.0}$
	14	0.9736	$2^{37.9}$	293	285	$2^{52.4}$	$2^{29.0}$
Well-Pooled MMT [33]	13	0.9629	$2^{39.1}$	315	303	$2^{52.6}$	$2^{25.2}$

*The complexity of MMT is $2^{0.05363e}$.

⋆The complexity of BJMM is $2^{0.04934e}$.

Comparison with Other Methods In this section, we compare our attack with GAD and ISD methods. The comparison is summarized in Table 3. The results indicate that our attack outperforms GAD significantly. In addition, SubGauss and ISD solvers including MMT, BJMM and Well-Pooled MMT achieve comparable performance, with only a marginal advantage in favor of one algorithm when differences occur. We next present the details of the complexity evaluation.

According to the Proposition 4 in [28], the GAD can recover the seed of $\mathsf{XOR}_a - \mathsf{MAJ}_b$ in time complexity $2^{n^{1-(s-1)/(\lceil \frac{b}{2} \rceil + 1)}}$. By substitution, the complexity of GAD on the instance is 2^{131}. By contrast, the grouping methods achieve practical attack on a small cluster of CPU cores. To simplify our comparison, we focus on the time complexity and the data complexity, omitting the discussion of memory complexity. In practical application, the memory complexity is crucial while Gauss and SubGauss solvers need small memory costs.

Then we evaluate the complexities of the attacks equipped with MMT and BJMM solvers. As discussed in Sec. 1, we consider the more competitive *half distance decoding* version, which strengthens our attack. ISD solvers were designed to solve an (n', k', d, w') decoding problem, where n' denotes the codeword length, k' denotes the message length, d denotes the minimal

distance of the code and w' denotes the largest possible number of errors in the code. To guarantee successful decoding, $w' < d/2$. We simplify to adopt the Gilbert-Varshamov bound [45] $d \approx n' H_2^{-1}(1 - \frac{k'}{n'})$, which is usually achieved when n' is sufficiently large. Here $H_2(x) := -x\log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function and H_2^{-1} is the inverse of H_2. To evaluate the time complexity of the ISD solvers, we simplify to model the LPN instance as $(e, n-r, d, e(1-p))$ code, where e denotes the number of used equations in each group (and is also the code length for the ISD solver) and $d \approx e \cdot H_2^{-1}(1-(n-r)/e)$. Then the evaluation is reduced to finding approximate r to minimize e with the constraint $w' < d/2$, i.e., $1 - p < H_2^{-1}(1-(n-r)/e)$, since e directly dominates the computational complexity. The complexities for different r can be obtained by heuristic for MMJ and BJMM.

Finally, we evaluate the complexity of the Well-Pooled MMT algorithm. As analyzed in [33], the complexity of the LPN solver with Well-Pooled MMT algorithm over k variables, where each noisy equation holds with probability p, is

$$2^{\frac{\log \frac{1}{1-\frac{7}{6}(1-p)}}{\log \frac{12}{5} + \log \frac{1}{1-\frac{7}{6}(1-p)}} k}, \tag{12}$$

with the same required number of equations. By trials of r, we choose $r = 13$. Substituting $p = 0.9629$ and $k = n - r = 243$ into Eq. (12), the complexity of LPN solver with ISD algorithm is $2^{11.7}$ with $2^{11.7}$ noisy equations in each group, hence $m = 2^{39.1}$ output bits are used by (3). Hence the grouping complexity and the solving complexity are $\frac{\binom{n}{r}}{\binom{n/2}{r}} \cdot 2^{39.1} = 2^{52.6}$ and $\frac{\binom{n}{r}}{\binom{n/2}{r}} \cdot 2^{11.7} = 25.2$, respectively.

Experiment We implement the GAS attack and publish our implementation (C++) at `https://github.com/Analytic233/GAS-attack`.

We execute our code on a single commodity desktop PC equipped with an Intel i5-13600KF processor (14 cores). For reproducibility, our C++ implementation utilized the standard library's pseudorandom number generator, seeded with the SHA-256 hash of the string "EUROCRYPT2026". In our execution, based on the Table 3, we selected the optimal rank parameter $r = 12$. The experiment began with a preliminary data generation phase, during which $2^{33.2}$ output bits are generated and stored along with their corresponding input indices to disk. The main attack was then parallelized across the 14 CPU cores. Each core independently ran the `Equation Grouping` algorithm to find candidate groups. For each group found, we performed Gaussian elimination on a random subset of 244 equations to derive a potential seed, which was then verified. To account for the probabilistic nature of this selection, the solving and verification process was repeated up to an expected $1/0.9495^{244} \approx 309904$ times per group. The correct seed was recovered after evaluating around 2180 (out of expected 8488) groups, costing 71 hours.

Furthermore, we generated keys from an additional set of seeds and conducted multi-instance experiments. The results are listed in table 4. Additionally, we provide a simple toy example in our repository for a quick demonstration and verification of our attack. This example is configured with a reduced parameter set ($N = 60$, $a = 5$, $b = 15$) and the attack fixes $r = 5$ bits. It performs the complete attack, from grouping to solving, and successfully recovers the 60-bit secret key, typically in less than 30 seconds on a single CPU core.

Table 4: Some experimental results for different seeds.

Seed	**Groups**	**Time**	**Cores**
SHA256(EUROCRYPTO2025)	500	4.7h	32
SHA256(CRYPTO2025)	1440	44h	14
SHA256(EUROCRYPT2026)	2180	71h	14

Note that our implementation is less optimized. In this attack, the Hamming weight of each correlation equation is $a = 10$ while there are $n - r = 244$ unknowns, i.e., the equations are sparse. Hence, the attack can be further optimized by techniques for solving sparse linear equations and is feasible on a single PC in practice. The similar techniques also applies to the FiLIP cipher.

As is discussed in Section 3.3, we compute the holding probability of each noisy equation by Equ. (1) with $w = n/2 = 128$, which is the worst case. The deviation from $n/2$ could help reduce the attack complexity. For example, if $w = 129$, substituted into Equ. (1), the probability p is 0.9560. Hence, the grouping complexity and solving complexity are reduced by a factor $\frac{\binom{129}{12}}{\binom{128}{12}} = \frac{129}{117} = 2^{0.14}$ and $2^{0.14} \cdot \left(\frac{0.956}{0.9495}\right)^{244} = 2^{2.5}$, respectively.

5 Key-recovery Attacks on FiLIP Instances

In this section, we further extend our attack to the FiLIP cipher [56].

The encryption scheme of FiLIP is as follows. Given a secret key x of length n, the keystream y is generated as depicted in Figure 5, similarly to Goldreich's PRGs discussed earlier, with the difference that the input to the predicate P is the sum of the key bits $x[\sigma_i]$ and a publicly known whitening noise vector w_i. That is, each output bit y_i of FiLIP is computed as

$$y_i = P\big(x[\sigma_i] + w_i\big).$$

Our attack focuses on the instances proposed in [39], which use smaller key sizes and shorter outputs, along with extensive cryptanalysis, thus reducing both computation and storage requirements compared to the instances in [56]. In [39], the authors introduced two types of FiLIP instances based on the XOR-THR

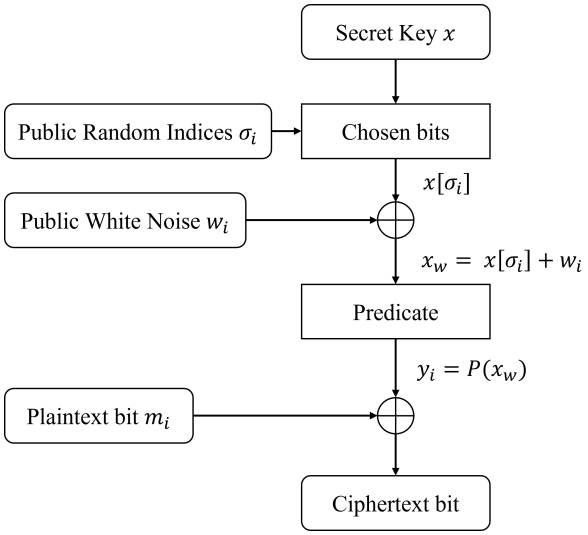


Fig. 5: Encryption Scheme of FiLIP

predicate and the XOR-THR-THR predicate, respectively. The λ-bit security is claimed to guarantee to restrict the output length within $2^{\lambda/2}$.

We provide attacks on both types of instances in Sections 5.1 and 5.2, respectively, i.e., recovering the secret key x given the predicate P, outputs y, indices Σ, and the noise vectors $w = (w_1, w_2 \ldots, w_m)$. As discussed in Section 3.3, the SubGauss solver outperforms the Gauss solver when the output length is relatively large, confirmed by the instances in Section 4. ISD solvers work with competitive performance with the SubGauss solver. Hence, for simplicity, we apply the SubGauss solver in this section.

5.1 Attack on FiLIP Instantiated with XOR-THR

For FiLIP instantiated with the XOR_a-$\mathsf{THR}_{t,b}$ predicate, each output y_i can be expressed as

$$y_i = \mathsf{XOR}_a\big(x[\sigma_{i,1}] + w_i[1{:}a]\big) + \mathsf{THR}_{t,b}\big(x[\sigma_{i,2}] + w_i[a{+}1{:}a{+}b]\big),$$

where w_i represents the noise bits added to the input, $\sigma_{i,1}$ and $\sigma_{i,2}$ are disjoint sets.

The main idea of the attack remains the same as in Section 3. However, due to the whitening (i.e., the addition of noise bits), two differences have emerged: (1) the inputs to the predicate may differ even when the indices are the same, and (2) when the common input bits are all 1s, the remaining input bits can be treated as 0-1 balanced due to the whitening, while in Section 3 the remaining part necessarily has a smaller Hamming weight because the seed is a fixed string.

The second difference leads to a simpler property, as stated in Proposition 2.

Proposition 2. *For the FiLIP cipher employing the predicate* XOR_a-$\mathsf{THR}_{t,b}$*, by fixing* $r \leq t$ *bits in the* THR *part to ones with the other bits uniformly chosen from* $\{0,1\}^{b-r}$*, a noisy equation* $y_i = \mathsf{XOR}_a(x[\sigma_{i,1}])+1$ *is derived with probability*

$$\frac{1}{2^{b-r}} \sum_{i=t}^{b-r} \binom{b-r}{i}. \tag{13}$$

To address the first issue, we propose a new grouping and solving attack. The main idea is to guess a sufficiently large number of (chosen arbitrarily) bits of the secret key. For each guess, added by the whitening noise, the corresponding inputs taken into the MAJ part are known. If at least r bits are 1s, then the noisy equations derived by Proposition 2 are collected and solved.

The attack consists of two phases, namely *data filtering phase* and *guess and solve phase*.

In the *data filtering* phase, First, fix two parameters l and r with $l > r$. Randomly choose a set $\mathcal{L}$ containing l indices from the key indices $[1, n]$. For each y_i, we append the tuple (y_i, σ_i) to the collected set $\mathcal{B}$ if the intersection size satisfies $r \leq |\sigma_{i,2} \cap \mathcal{L}| \leq l$. This phase rules out the output data that have fewer than r common indices with the predetermined set, avoiding the subsequent processing of unrelated data, thereby reducing computational costs.

In the *guess and solve* phase, we attempt to guess the values of $x[\mathcal{L}]$. For each guess, we initialize an empty group $\mathcal{G}$. For each tuple (y_i, σ_i) in $\mathcal{B}$, we consider the corresponding noisy equation $y_i' = \mathsf{XOR}_a\big(x[\sigma_{i,1}] + w_i[1{:}a]\big) + 1$, and append it to $\mathcal{G}$ if there exists a subset $\mathcal{R}_i \subseteq \sigma_{i,2} \cap \mathcal{L}$ with $|\mathcal{R}_i| \geq r$ such that $x[\mathcal{R}_i] + w_i[\mathrm{pos}(\mathcal{R}_i, \sigma_i)] = 1$. Here, $\mathrm{pos}(\mathcal{R}_i, \sigma_i)$ maps each index in $\mathcal{R}_i$ to the corresponding position in w_i, ranging from $a+1$ to $a+b$. In other words, for each index $j \in \mathcal{R}_i$, the key bit x_j and the corresponding noise bit of index $\mathrm{pos}(j, \sigma_i)$ satisfy $x_j + w_i[\mathrm{pos}(j, \sigma_i)] = 1$. Hence, a group $\mathcal{G}$ of noisy equations is derived for each guess and can be solved by the SubGauss solver and solutions can be verified by substituting into the FiLIP cipher. For the correct guess, the derived group of noisy equations is highly biased and successfully solved, then we obtain the key.

The attack is detailed in Algorithm 4.

Analysis of Complexity and Success Rate It is clear that the complexity of *data filtering* is m. For $u \in [r, \min(l, b)]$, u out of the l bits of $\mathcal{L}$ appear in the THR predicate with the probability

$$p_u = \frac{\binom{l}{u}\binom{n-l}{b-u}}{\binom{n}{b}}. \tag{14}$$

After the *data filtering* step, the set $\mathcal{B}$ is of size $\sum_{u=r}^{l} mp_u$.

It is clear that the complexity of *guess and solve* step is

$$2^l \left(\sum_{u=r}^{l} mp_u + T_s \right),$$

Algorithm 4 Attack on FiLIP Instantiated with XOR_a-$\mathsf{THR}_{t,b}$

Input: Number of fixing bits r, larger range l.
$e \leftarrow \sum_{u=r}^{l} \sum_{v=r}^{u} mp_u 2^{-u} \binom{u}{v}, p \leftarrow \frac{1}{2^{b-r}} \sum_{i=t-r}^{b-r} \binom{b-r}{i}$
Data Filtering Phase:
$\mathcal{B} \leftarrow \emptyset$, randomly chosen l indices forming $\mathcal{L}$
for $i = 1$ **to** m **do**
  if $r \leq |\sigma_{i,2} \cap \mathcal{L}| \leq l$ **then**
    $\mathcal{B}$.append (y_i, σ_i)
  end if
end for
Guess and Solve Phase:
for a guess $x[\mathcal{L}] \in \{0,1\}^l$ **do**
  $\mathcal{G} \leftarrow \emptyset$
  for $(y_i, \sigma_i) \in \mathcal{B}$ **do**
    if exists $\mathcal{R}_i \subseteq \sigma_{i,2} \cap \mathcal{L}$ with $|\mathcal{R}_i| \geq r$ such that $x[\mathcal{R}_i] + w_i[\mathrm{pos}(\mathcal{R}_i, \sigma_i)] = 1$ **then**
      $\mathcal{G}$.append $(y_i = \mathsf{XOR}(x[\sigma_{i,1}] + w_i[1:a]) + 1)$
    end if
  end for
  $X \leftarrow$ `System Solving` $(\mathcal{G}, n - r, ep)$
  for $x \in X$ **do**
    if $P(x[\sigma_i] + w_i | x[\mathcal{L}]) = y_i$ for all $i \in [1, m]$ **then**
      return x
    end if
  end for
end for
return $\perp$

where T_s denotes the complexity of solving a group of noisy equations and verifying solutions. In order to evaluate T_s, we first evaluate the expected number of noisy equations in each group e. For a tuple with u ($r \leq u \leq l$) bits in $\mathcal{L}$, at least r bits taken into the THR predicate are all ones after the whitening with probability

$$2^{-u} \sum_{v=r}^{u} \binom{u}{v}. \tag{15}$$

Hence, the expected number of noisy equations in each group is

$$e = \sum_{u=r}^{l} mp_u 2^{-u} \sum_{v=r}^{u} \binom{u}{v}. \tag{16}$$

Similar to the analysis in Section 3.3, the complexity of the solving phase is

$$T_s = \begin{cases} \frac{1}{p^{n-l}}(n-l)^\omega, & e \geq (n-l)/p, \\ \frac{2^{n-l-ep}}{p^{ep}}(ep)^\omega, & e < (n-l)/p. \end{cases}$$

Hence, the total complexity of the attack on FiLIP instantiated with XOR-THR predicate is

$$T_{xt} = 2^l m \sum_{u=r}^{l} p_u + \frac{2^{n-\min(n-l,ep)}}{p^{\min(ep,n-l)}} n^\omega \tag{17}$$

As discussed in Section 2.3, Gaussian elimination has been extensively studied and has witnessed many advances, especially for solving sparse equations [24,65,48]. Here, we use the naive implementation ($\omega = 3$) to evaluate the bit complexity, i.e., $T_{bit} = T_s \cdot (n-l)^3$ to demonstrate the efficiency. We use $2(a+b)$ to estimate the bit operations for each encryption, i.e., one $a+b$ counting the bit operations for whitening and the other $a+b$ counting the bit operations for computing the $\mathsf{XOR} - \mathsf{MAJ}$ predicate. Hence, the encryption complexity is $T_{enc} = \frac{T_{bit}}{2(a+b)} = \frac{T_s \cdot (n-l)^3}{2(a+b)}$.

Differently with the attack in Section 3, we guess a set of bits rather than random 2^r trials. The correct answer of the l bits must not be overlooked. The attack succeeds only when the SubGauss solver succeeds in the group with correct guess. Consistent with the analysis in Section 3.3, the SubGauss solver succeeds and hence the attack shown in Algorithm 4 succeeds with the probability $1 - \frac{1}{E}$, where E denotes the base of natural logarithm.

Table 5: Attack Parameters for FiLIP with $\mathsf{XOR_a\text{-}THR_{t,b}}$. D denotes the consumed data complexity. e and ep denote the noisy equations and the correct ones on average in each group, respectively. T_s denotes the number of Gaussian eliminations. T_{bit} denotes the number of bit operaions and T_{enc} denotes the equivalent number of encryptions.

a	t	b	n	r	l	p	D	e	ep	T_{eg}	T_s	T_{bit}	T_{enc}	2^λ
40	52	104	530	16	29	0.9653	$2^{39.5}$	518	501	$2^{52.2}$	$2^{54.5}$	$2^{79.1}$	$2^{70.9}$	2^{80}
100	24*	44	982	10	34	0.9878	$2^{39.8}$	959	948	$2^{53.3}$	$2^{50.7}$	$2^{80.4}$	$2^{72.2}$	2^{80}
65	32	63	2560	15	60	0.9853	$2^{64.0}$	2537	2500	$2^{85.7}$	$2^{113.2}$	$2^{147.1}$	$2^{139.1}$	2^{128}
58	35	70	4096	14	68	0.9780	$2^{63.8}$	4118	4028	$2^{93.5}$	$2^{197.4}$	$2^{233.3}$	$2^{225.3}$	2^{128}
80	260	520	1200	41	56	0.9726	$2^{63.0}$	1176	1144	$2^{100.8}$	$2^{101.9}$	$2^{132.4}$	$2^{122.2}$	2^{128}
70	93	186	1499	26	60	0.9838	$2^{63.8}$	1462	1439	$2^{93.7}$	$2^{94.0}$	$2^{125.5}$	$2^{116.5}$	2^{128}
65	96	191	1461	27	63	0.9827	$2^{63.3}$	1422	1398	$2^{97.3}$	$2^{98.3}$	$2^{129.6}$	$2^{120.6}$	2^{128}
70	61	122	1777	21	61	0.9860	$2^{63.5}$	1740	1716	$2^{91.1}$	$2^{96.0}$	$2^{128.2}$	$2^{119.6}$	2^{128}
160	48	96	1987	19	64	0.9890	$2^{63.9}$	1944	1923	$2^{92.7}$	$2^{94.7}$	$2^{127.4}$	$2^{118.4}$	2^{128}

*$\mathsf{THR}_{24,44}(x)$ can be treated as a dual threshold predicate for 0, i.e., $\mathsf{THR}_{24,44}(x) = 0$ if there are more than or equal to 21 zeros. The corresponding equation is flipped, i.e., append the equations $y_i = \mathsf{XOR}(x[\sigma_{i,1}] + w_i[1:a]))$ rather than $y_i = \mathsf{XOR}(x[\sigma_{i,1}] + w_i[1:a]) + 1$ (line 14) in Algorithm 4.

The parameters r and l are chosen to minimize the complexity T_{xt} in Eq. (17). We provide by heuristic the results, as shown in Table 5, and our codebase includes a simple simulation to validate these instances.

The results demonstrate the efficiency of our attacks for most instances, especially in terms of Gaussian eliminations. For example, FiLIP cipher with XOR_{40}-$\mathsf{THR}_{52,104}$ can be broken by no more than $2^{54.5}$ calls to Gaussian elimination, while the key size is 530 and security assumption is $\lambda = 80$. Though the bit complexity reaches up to $2^{79.1}$, almost equal to the claimed security level, we believe our attack is much more efficient than the estimated bit complexity in practice. Most modern CPUs support SIMD instructions such that a row vector of 256 bits (some even support 512 bits) can be executed in a clock cycle. Hence, a row addition in this instance can be finished within two cycles. For a cluster of 1000000 CPU cores, each with 2GHz main frequency, the attack can be performed in $\frac{2^{54.5-31} \cdot 501^2 * 2}{3600*24*30*1000000} = 2.3$ months. Hence, the attack is feasible in practice. The analysis is applicable to the other cases.

5.2 Attack on FiLIP Instantiated with XOR-THR-THR

In the same paper [39], another instance with the XOR-THR-THR predicate was proposed, defined as

$$\mathsf{XOR}_a\text{-}\mathsf{THR}_{t_1,b_1}\text{-}\mathsf{THR}_{t_2,b_2}(x[\sigma_i]) = \mathsf{XOR}_a\big(x[\sigma_{i,1}]\big) + \mathsf{THR}_{t_1,b_1}\big(x[\sigma_{i,2}]\big) + \mathsf{THR}_{t_2,b_2}\big(x[\sigma_{i,3}]\big).$$

Similarly, by fixing partial bits, a noisy equation can be derived, as justified in Proposition 3.

Proposition 3. *For the FiLIP cipher employing the predicate* $\mathsf{XOR}_a\text{-}\mathsf{THR}_{t_1,b_1}\text{-}\mathsf{THR}_{t_2,b_2}(x[\sigma_i])$*, by fixing* r_1 *bits in the* THR_{t_1,b_1} *predicate and* r_2 *bits in the* THR_{t_2,b_2} *predicate to 1s, a noisy equation* $y_i = \mathsf{XOR}_a\big(x[\sigma_{i,1}]\big)$ *is derived with probability* $\frac{1+(2p_1-1)(2p_2-1)}{2}$*, where*

$$p_1 = \frac{1}{2^{b_1-r_1}} \sum_{u_1=t_1-r_1}^{b_1-r_1} \binom{b_1-r_1}{u_1},$$

$$p_2 = \frac{1}{2^{b_2-r_2}} \sum_{u_2=t_2-r_2}^{b_2-r_2} \binom{b_2-r_2}{u_2}.$$

Proof. Proposition 3 can be proved by combining Proposition 2 and the well-known piling-up lemma[60].

The attack on the FiLIP cipher instantiated with the XOR-THR-THR predicate is similar to the attack on the XOR-THR instance. The main idea is to guess a sufficiently large number of bits of the key. If there are more than r_1 1s in the THR_{t_1,b_1} part and r_2 1s in the THR_{t_1,b_1} part, then the noisy equations derived by Proposition 3 are collected and solved.

The attack involves two phases similar to the previous case, i.e., the *data filtering* phase and the *guess and solve* phase.

In the *data filtering* phase, arbitrarily choose a set $\mathcal{L}$ of l indices and select parameters (l, r_1, r_2) such that $r_1 + r_2 \leq l$. For each output bit y_i, append the tuple (y_i, σ_i) to the collected set $\mathcal{B}$ if

$$r_1 \leq |\sigma_{i,2} \cap \mathcal{L}| \leq l \quad \text{and} \quad r_2 \leq |\sigma_{i,3} \cap \mathcal{L}| \leq l.$$

Similar to the analysis for XOR-THR instances, this phase can reduce computational costs by ruling out unrelated data.

In the *guess and solve* phase, we guess the l bits in $\mathcal{L}$. For each guess, initialize an empty equation group $\mathcal{G}$. For each tuple (y_i, σ_i) in $\mathcal{B}$, we consider the equation $y_i' = \mathsf{XOR}_a\big(x[\sigma_{i,1}] + w_i[1{:}a]\big)$ and append it to $\mathcal{G}$ if there exist subsets $\mathcal{R}_{1,i} \subseteq \sigma_{i,2} \cap \mathcal{L}$ and $\mathcal{R}_{2,i} \subseteq \sigma_{i,3} \cap \mathcal{L}$, with $|\mathcal{R}_{1,i}| \geq r_1$ and $|\mathcal{R}_{2,i}| \geq r_2$, such that

$$x[\mathcal{R}_{1,i}] + w_i[\mathrm{pos}(\mathcal{R}_{1,i}, \sigma_i)] = 1,$$
$$x[\mathcal{R}_{2,i}] + w_i[\mathrm{pos}(\mathcal{R}_{2,i}, \sigma_i)] = 1.$$

Here, $\mathrm{pos}(\mathcal{R}_{1,i}, \sigma_i)$ maps each index in $\mathcal{R}_{1,i}$ to the corresponding position for w_i, ranging from $a+1$ to $a+b_1$ and $\mathrm{pos}(\mathcal{R}_{2,i}, \sigma_i)$ maps each index in $\mathcal{R}_{2,i}$ to the corresponding position for w_i, ranging from $a+b_1+1$ to $a+b_1+b_2$. Apply the SubGauss solver to the group $\mathcal{G}$ and verify the solutions by substituting them back into the FiLIP cipher. For the correct guess, the derived group of noisy equations is highly biased and successfully solved, then we obtain the key.

The attack details are provided in Algorithm 5.

Complexity and Success Rate Analysis It is clear that the complexity of *data filtering* is m. Similar to the previous analysis, u_1 out of the l bits appear in the THR_{t_1,b_1} predicate and u_2 out of the remaining $l-u_1$ bits appear in the THR_{t_2,b_2} predicate with the probability

$$p_{u_1,u_2} = \frac{\binom{l}{u_1}\binom{l-u_1}{u_2}\binom{n-l}{b_1-u_1}\binom{n-l-(b_1-u_1)}{b_2-u_2}}{\binom{n}{b_1}\binom{n-b_1}{b_2}}, \tag{18}$$

After the *data filtering* phase, the set $\mathcal{B}$ is of size $\sum_{u_1=r_1}^{l-r_2}\sum_{u_2=r_2}^{l-u_1} mp_{u_1,u_2}$.

It is clear that the complexity of *guess and solve* step is

$$2^l\Big(\sum_{u_1=r_1}^{l-r_2}\sum_{u_2=r_2}^{l-u_1} mp_{u_1,u_2} + T_s\Big),$$

where T_s denotes the complexity of solving a group of noisy equations and verifying solutions. In order to evaluate T_s, we need to evaluate the expected number of equations in each group e. For a tuple with u_1 ($r_1 \leq u_1 \leq l$) bits in the THR_{t_1,b_1} predicate and u_2 bits in the THR_{t_2,b_2} predicate, at least r_1 bits taken into the THR_{t_1,b_1} predicate and r_2 bits taken into the THR_{t_2,b_2} predicate are all ones with the probability

$$2^{-u_1}\sum_{v_1=r_1}^{u_1}\binom{u_1}{v_1}2^{-u_2}\sum_{v_2=r_2}^{u_2}\binom{u_2}{v_2}.$$

Algorithm 5 Attack on FiLIP Instantiated with XOR-THR-THR

Input: Number of fixing bits r_1, r_2, larger range l.

$e \leftarrow \sum_{u_1=r_1}^{l-r_2} \sum_{u_2=r_2}^{l-u_1} mp_{u_1,u_2} 2^{-u_1-u_2} \sum_{v_1=r_1}^{u_1} \binom{u_1}{v_1} \sum_{v_2=r_2}^{u_2} \binom{u_2}{v_2}$
$p_1 \leftarrow \frac{1}{2^{b_1-r_1}} \sum_{u_1=t_1-r_1}^{b_1-r_1} \binom{b_1-r_1}{u_1}$
$p_2 \leftarrow \frac{1}{2^{b_2-r_2}} \sum_{u_2=t_2-r_2}^{b_2-r_2} \binom{b_2-r_2}{u_2}$
$p \leftarrow \frac{1+(2p_1-1)*(2p_2-1)}{2}$
Data Filtering Phase:
$\mathcal{B} \leftarrow \emptyset$, randomly chosen l indices forming $\mathcal{L}$
for $i = 1$ **to** m **do**
  if $r_1 \leq |\sigma_{i,2} \cap \mathcal{L}| \leq l$ and $r_2 \leq |\sigma_{i,3} \cap \mathcal{L}| \leq l$ **then**
    $\mathcal{B}$.append (y_i, σ_i)
  end if
end for
Guess and Solve Phase:
for each guess $x[\mathcal{L}] \in \{0,1\}^l$ **do**
  $\mathcal{G} \leftarrow \emptyset$
  if exists $\mathcal{R}_{1,i} \subseteq \sigma_{i,2} \cap \mathcal{L}$ with $|\mathcal{R}_{1,i}| \geq r_1$ such that $x[\mathcal{R}_{1,i}] + w_i[\text{pos}(\mathcal{R}_{1,i}, \sigma_i)] = 1$ and $\mathcal{R}_{2,i} \subseteq \sigma_{i,3} \cap \mathcal{L}$ with $|\mathcal{R}_{2,i}| \geq r_2$ such that $x[\mathcal{R}_{2,i}] + w_i[\text{pos}(\mathcal{R}_{2,i}, \sigma_i)] = 1$ **then**
    $\mathcal{G}$.append $(y_i = \mathsf{XOR}(x[\sigma_{i,1}] + w_i[1:a]) + 1)$
  end if
end for
$X \leftarrow$ `System Solving` $(\mathcal{G}, n-r, ep)$
for $x \in X$ **do**
  if $P(x[\sigma_i] + w_i | x[\mathcal{L}]) = y_i$ for all $i \in [1, m]$ **then**
    return x
  end if
end for
return $\perp$

Hence, the expected number of noisy equations in each group for each guess is

$$e = \sum_{u_1=r_1}^{l-r_2} \sum_{u_2=r_2}^{l-u_1} mp_{u_1,u_2} 2^{-u_1-u_2} \sum_{v_1=r_1}^{u_1} \binom{u_1}{v_1} \sum_{v_2=r_2}^{u_2} \binom{u_2}{v_2}. \tag{19}$$

The total complexity of the attack on FiLIP instantiated with XOR-THR predicate is

$$T_{xtt} = \begin{cases} 2^l m \sum_{u_1=r_1}^{l-r_2} \sum_{u_2=r_2}^{l-u_1} p_{u_1,u_2} + \frac{2^l}{p^{n-l}}(n-l)^\omega, & e \geq (n-l)/p, \\ 2^l m \sum_{u_1=r_1}^{l-r_2} \sum_{u_2=r_2}^{l-u_1} p_{u_1,u_2} + \frac{2^{n-ep}}{p^{ep}}(ep)^\omega, & e < (n-l)/p. \end{cases} \tag{20}$$

Similar to the discussion for FiLIP instantiated with XOR − MAJ, the bit complexity is $T_{bit} = T_s \cdot (n-l)^3$. We use $2(a + b_1 + b_2)$ to estimate the bit

operations for each encryption, i.e., one $a + b_1 + b_2$ counting the bit operations for whitening and the other $a + b_1 + b_2$ counting the bit operations for computing the $\mathsf{XOR} - \mathsf{MAJ} - \mathsf{MAJ}$ predicate. Hence, the encryption complexity is

$$T_{enc} = \frac{T_{bit}}{2(a+b)} = \frac{T_s \cdot (n-l)^3}{2(a+b_1+b_2)}.$$

Identical to the attack in Section 5.1, the attack succeeds with the probability $1 - \frac{1}{E}$, where E denotes the base of natural logarithm.

The parameters r_1, r_2 and l are determined to minimize the complexity T_{xtt} in Equ. (20) and the results are shown in Table 6. The results demonstrate the efficiency of our attacks for most instances. For example, FiLIP cipher instantiated with $\mathsf{XOR}_{100}\text{-}\mathsf{THR}_{11,22}\text{-}\mathsf{THR}_{11,22}$ can be broken with no more than 2^{59} calls to Gaussian elimination, while the key size is 397 and the security assumption is $\lambda = 80$. The bit complexity can reach up to $2^{83.5}$, but the attack is still believed to be efficient in practice, similar to the analysis for FiLIP instantiated with $\mathsf{XOR} - \mathsf{MAJ}$ predicates.

Table 6: Attack Parameters for FiLIP Instantiated with $\mathsf{XOR}_{\mathsf{a}}\text{-}\mathsf{THR}_{\mathsf{t}_1,\mathsf{b}_1}\text{-}\mathsf{THR}_{\mathsf{t}_2,\mathsf{b}_2}$. D denotes the consumed data complexity. e and ep denote the noisy equations and the correct ones on average in each group, respectively. T_s denotes the number of Gaussian eliminations. T_{bit} denotes the number of bit operations. T_{enc} denotes the equivalent number of encryptions.

a	t_1	b_1	t_2	b_2	n	r_1	r_2	l	p	D	e	ep	T_{eg}	T_s	T_{bit}	T_{enc}	2^λ
40	26	52	26	52	360	11	11	42	0.9422	$2^{39.2}$	337	318	$2^{68.4}$	$2^{69.3}$	$2^{94.2}$	$2^{86.0}$	2^{80}
100	11	22	11	22	397	7	7	40	0.9655	$2^{38.8}$	369	357	$2^{61.1}$	$2^{58.1}$	$2^{83.5}$	$2^{80.5}$	2^{80}
54	19	38	19	38	6500	7	7	109	0.8616	$2^{63.8}$	7417	6391	$2^{135.1}$	$2^{1482.7}$	$2^{1520.6}$	$2^{1512.6}$	2^{128}
54	22	45	23	45	3072	9	10	111	0.9255	$2^{63.9}$	3199	2961	$2^{140.1}$	$2^{441.6}$	$2^{476.2}$	$2^{468.0}$	2^{128}
70	30	61	31	61	841	13	14	61	0.9659	$2^{63.9}$	807	780	$2^{94.6}$	$2^{100.0}$	$2^{128.8}$	$2^{120.2}$	2^{128}
65	47	95	48	96	830	16	18	66	0.9642	$2^{63.9}$	792	764	$2^{104.5}$	$2^{106.2}$	$2^{134.9}$	$2^{125.9}$	2^{128}
70	46	93	47	93	736	16	18	62	0.9625	$2^{63.3}$	700	674	$2^{100.4}$	$2^{99.2}$	$2^{127.4}$	$2^{118.4}$	2^{128}
60	26	53	27	53	1229	11	13	74	0.9598	$2^{63.8}$	1203	1155	$2^{105.7}$	$2^{142.3}$	$2^{172.8}$	$2^{164.4}$	2^{128}
160	24	48	24	48	913	12	12	62	0.9716	$2^{63.9}$	875	851	$2^{93.5}$	$2^{97.4}$	$2^{126.6}$	$2^{117.6}$	2^{128}

6 Conclusion

In this paper, we introduced the Group-and-Solve attack on Goldreich's PRGs based on $\mathsf{XOR}\text{-}\mathsf{THR}$ predicates. We applied our attack to the XOR-MAJ family proposed at STOC 2016 and to an instance from Eurocrypt 2024, the latter of which we also implemented on a small CPU cluster. Furthermore, we extended our method to the FiLIP stream cipher instantiated with THR-related predicates, successfully breaking most instances within the claimed complexity bounds.

Regarding countermeasures, our analysis shows that the attack becomes less effective when $n > (b - s + 1)^s + s$, as noted just before Section 4. Meanwhile, the STOC 2016 shows that algebraic attacks become ineffective when $b > 36s$. Hence, when the parameters (n, b, s) satisfy both inequalities, XOR-MAJ is expected to resist both attacks. However, for many choices of (n, s), there is no feasible value of b that satisfies both constraints. For example, for $n = 256$ and $s = 5$ in the Eurocrypt 2024 instance, no value of b satisfies both inequalities simultaneously. This suggests that relying solely on parameter tuning is not a viable countermeasure for XOR-MAJ. We therefore believe that designing new predicates is a more promising direction for future work.

References

1. Agarwal, A., Baum, C., Braun, L., Scholl, P.: Low-bandwidth mixed arithmetic in vole-based zk from low-degree prgs. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 396–426. Springer (2025)
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. pp. 430–454 (2015)
3. Applebaum, B.: Pseudorandom generators with long stretch and low locality from random local one-way functions. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012. pp. 805–816 (2012)
4. Applebaum, B., Bogdanov, A., Rosen, A.: A dichotomy for local small-bias generators. In: Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings. pp. 600–617 (2012)
5. Applebaum, B., Damgård, I., Ishai, Y., Nielsen, M., Zichron, L.: Secure arithmetic computation with constant computational overhead. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 223–254. Springer (2017)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in NC^0. Comput. Complex. **17**(1), 38–69 (2008)
7. Applebaum, B., Konstantini, N.: Actively secure arithmetic computation and VOLE with constant computational overhead. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 190–219. Springer (2023)
8. Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016. pp. 1087–1100 (2016)

9. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM **45**(3), 501–555 (1998)
10. Arora, S., Safra, S.: Probabilistic checking of proofs; A new characterization of NP. In: 33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992. pp. 2–13 (1992)
11. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM **45**(1), 70–122 (1998)
12. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 1–18. Springer (2001)
13. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6:1–6:48 (2012)
14. Beauchamp, K.G., Debnath, L.: Walsh functions and their applications. IEEE Transactions on Systems, Man, and Cybernetics **9**(1), 67–67 (1979)
15. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 520–536. Springer (2012)
16. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: Ball-collision decoding. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 743–760. Springer (2011)
17. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA. pp. 435–440 (2000)
18. Bogdanov, A., Qiao, Y.: On the security of Goldreich's one-way function. Computational Complexity **21**(1), 83–127 (2012). `https://doi.org/10.1007/s00037-011-0034-0`
19. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Orrù, M.: Homomorphic secret sharing: Optimizations and applications. In: Thuraisingham, B., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 2105–2122. ACM (2017)
20. Brakerski, Z., Lyubashevsky, V., Vaikuntanathan, V., Wichs, D.: Worst-case hardness for LPN and cryptographic hashing via code smoothing. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 619–635. Springer (2019)
21. Bui, D., Couteau, G., Meyer, P., Passelègue, A., Riahinia, M.: Fast public-key silent OT and more from constrained naor-reingold. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich,

Switzerland, May 26-30, 2024, Proceedings, Part VI. Lecture Notes in Computer Science, vol. 14656, pp. 88–118. Springer (2024)
22. Carrijo, J., Tonicelli, R., Imai, H., Nascimento, A.C.A.: A novel probabilistic passive attack on the protocols HB and hb^{+}. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **92-A**(2), 658–662 (2009)
23. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA. pp. 151–158 (1971)
24. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. In: Proceedings of the nineteenth annual ACM symposium on Theory of computing. pp. 1–6 (1987)
25. Courtois, N.T.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 176–194. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_11, https://doi.org/10.1007/978-3-540-45146-4_11
26. Courtois, N.T., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 392–407. Springer (2000)
27. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656, pp. 345–359. Springer (2003)
28. Couteau, G., Dupin, A., Méaux, P., Rossi, M., Rotella, Y.: On the concrete security of goldreich's pseudorandom generator. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 96–124. Springer (2018)
29. Cryan, M., Miltersen, P.B.: On pseudorandom generators in NC. In: Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings. pp. 272–284 (2001)
30. Dinur, I.: Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 374–403. Springer (2021)
31. Dinur, I.: Improved algorithms for solving polynomial systems over GF(2) by multiple parity-counting. In: Marx, D. (ed.) Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021. pp. 2550–2564. SIAM (2021)
32. Ducas, L., Esser, A., Etinski, S., Kirshanova, E.: Asymptotics and improvements of sieving for codes. In: Joye, M., Leander, G. (eds.) Advances in Cryptology -

EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI. Lecture Notes in Computer Science, vol. 14656, pp. 151–180. Springer (2024)
33. Esser, A., Kübler, R., May, A.: LPN decoded. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 486–514. Springer (2017)
34. Faugère, J.C.: A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation. p. 75–83. ISSAC '02, Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/780506.780516, https://doi.org/10.1145/780506.780516
35. Faugére, J.C.: A new efficient algorithm for computing gröbner bases (f4). Journal of Pure and Applied Algebra **139**(1), 61–88 (1999). https://doi.org/https://doi.org/10.1016/S0022-4049(99)00005-5, https://www.sciencedirect.com/science/article/pii/S0022404999000055
36. Fossorier, M.P.C., Mihaljevic, M.J., Imai, H., Cui, Y., Matsuura, K.: A novel algorithm for solving the LPN problem and its application to security evaluation of the HB protocol for RFID authentication. IACR Cryptol. ePrint Arch. p. 197 (2006)
37. Fraenkel, A.S., Yesha, Y.: Complexity of solving algebraic equations. Inf. Process. Lett. **10**(4/5), 178–179 (1980)
38. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
39. Gérard, F., Gini, A., Méaux, P.: Toolip: How to find new instances of filip cipher with smaller key size and new filters. In: Vaudenay, S., Petit, C. (eds.) Progress in Cryptology - AFRICACRYPT 2024 - 15th International Conference on Cryptology in Africa, Douala, Cameroon, July 10-12, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14861, pp. 21–45. Springer (2024)
40. Goldreich, O.: Candidate one-way functions based on expander graphs. Electronic Colloquium on Computational Complexity (ECCC) **7**(90) (2000)
41. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: Mpc-friendly symmetric key primitives. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 430–443. ACM (2016)
42. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 1–20. Springer (2014)
43. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. J. Cryptol. **33**(1), 1–33 (2020)
44. Guo, Q., Johansson, T., Nguyen, V.: A new sieving-style information-set decoding algorithm. IEEE Trans. Inf. Theory **70**(11), 8303–8319 (2024)
45. Guruswami, V.: Introduction to Coding Theory. Lecture Notes (2010)
46. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM

Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 433–442. ACM (2008)
47. Joux, A., Vitse, V.: A crossbred algorithm for solving boolean polynomial systems. In: Kaczorowski, J., Pieprzyk, J., Pomykala, J. (eds.) Number-Theoretic Methods in Cryptology - First International Conference, NuTMiC 2017, Warsaw, Poland, September 11-13, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10737, pp. 3–21. Springer (2017)
48. Kaltofen, E., David Saunders, B.: On wiedemann's method of solving sparse linear systems. In: International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes. pp. 29–38. Springer (1991)
49. Levieil, É., Fouque, P.: An improved LPN algorithm. In: Prisco, R.D., Yung, M. (eds.) Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4116, pp. 348–359. Springer (2006)
50. Levin, L.A.: Universal sequential search problems. Problemy peredachi informatsii **9**(3), 115–116 (1973)
51. Lokshtanov, D., Paturi, R., Tamaki, S., Williams, R.R., Yu, H.: Beating brute force for systems of polynomial equations over finite fields. In: Klein, P.N. (ed.) Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19. pp. 2190–2202. SIAM (2017)
52. Lombardi, A., Vaikuntanathan, V.: Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 119–137. Springer (2017)
53. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3624, pp. 378–389. Springer (2005)
54. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 107–124. Springer (2011)
55. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 203–228. Springer (2015)
56. Méaux, P., Carlet, C., Journault, A., Standaert, F.: Improved filter permutators for efficient FHE: better instances and implementations. In: Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings. pp. 68–91 (2019)

57. Méaux, P., Journault, A., Standaert, F., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. pp. 311–343 (2016)
58. Mossel, E., Shpilka, A., Trevisan, L.: On e-biased generators in NC0. In: 44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings. pp. 136–145 (2003)
59. O'Donnell, R., Witmer, D.: Goldreich's PRG: evidence for near-optimal polynomial stretch. In: IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014. pp. 1–12 (2014)
60. Rueppel, R.A.: Analysis and design of stream ciphers (1986)
61. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014. pp. 475–484. ACM (2014)
62. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. SIAM J. Comput. **50**(3), 857–908 (2021)
63. Vidal, D., Delaplace, C., Ionica, S.: An analysis of the crossbred algorithm for the MQ problem. IACR Commun. Cryptol. **1**(3), 36 (2024)
64. Wei, C., Wu, C., Fu, X., Dong, X., He, K., Hong, J., Wang, X.: Preimage attacks on 4-round keccak by solving multivariate quadratic systems. In: Park, J.H., Seo, S. (eds.) Information Security and Cryptology - ICISC 2021 - 24th International Conference, Seoul, South Korea, December 1-3, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13218, pp. 195–216. Springer (2021)
65. Wiedemann, D.: Solving sparse linear equations over finite fields. IEEE transactions on information theory **32**(1), 54–62 (1986)
66. Yang, J., Guo, Q., Johansson, T., Lentmaier, M.: Revisiting the concrete security of goldreich's pseudorandom generator. IEEE Trans. Inf. Theory **68**(2), 1329–1354 (2022)
67. Zhang, B., Jiao, L., Wang, M.: Faster algorithms for solving LPN. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 168–195. Springer (2016)
68. Zichron, L.: Locally Computable Arithmetic Pseudorandom Generators. Master's thesis, Tel Aviv University (2017), `https://bennyapplebaum.sites.tau.ac.il/_files/ugd/f706bf_501515c9cd7744c498935684bd1648a2.pdf`