Bit-fixing Correlation Attacks on Goldreich's Pseudorandom Generators

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

[1] Harbin Institute of Technology, Shenzhen
{fuximing,liuchuanyi}@hit.edu.cn, 210310309@stu.hit.edu.cn
[2] The Chinese University of Hong Kong, Shenzhen
220019160@link.cuhk.edu.cn

Abstract. We introduce a powerful attack, termed the *bit-fixing correlation attack*, on Goldreich's pseudorandom generators (PRGs), specifically focusing on those based on the XOR-THR predicate. By exploiting the *bit-fixing correlation* property, we derive correlation equations with high bias by fixing certain bits. Utilizing two solvers to handle these high-bias correlation equations, we present inverse attacks on XOR-THR based PRGs within the complexity class NC^0.

We efficiently attack the XOR-MAJ challenges (STOC 2016), demonstrating that the XOR-MAJ predicate fails to be s-pseudorandom with n-bit security even for a stretch factor $s = 1$, where n is the size of the secret seed. For instance, a challenge of $n = 42$ and $s = 1$ can be broken using approximately 2^{28} calls to *Gaussian elimination*. We extend our attack to an instance used in constructing silent Oblivious Transfer (OT) protocols (Eurocrypt 2024), with $n = 256$. This attack can be realized with approximately 2^{29} calls to *Gaussian elimination*, and we have implemented this attack on a cluster of 32 CPU cores, successfully recovering the secret seed in 5.5 hours. Furthermore, we extend our results to general *Piecewise Symmetric Predicates* of the form XOR-X, showing that XOR-MAJ is far from well designed predicate against bit-fixing correlation attack.

With marginal modification, our attack can also be adapted to the FiLIP cipher instantiated with THR-related predicates, making it effective against most FiLIP instances. For example, the FiLIP cipher instantiated on XOR-THR with key size 982 can be broken using approximately 2^{51} calls to *Gaussian elimination*.

Based on these findings, we show that the traditional security assumptions for Goldreich's PRGs—namely, (a) $\Omega(s)$-resilience and (b) algebraic immunity—are insufficient to guarantee pseudorandomness or one-wayness.

Keywords: Golereich's PRGs, random local function, XOR-MAJ, correlation equation, FiLIP cipher

1 Introduction

A pseudorandom generator (PRG) is a fundamental cryptographic tool that takes a short random seed of length n as input and generates $m \gg n$ pseudorandom bits. Goldreich [16] introduced a family of one-way functions with a *locality* property: each output bit is generated by applying a fixed $d(n)$-ary predicate to a randomly selected subset of $d(n)$ input bits. Promising research on PRGs with locality have been inspired, thus are commonly named Goldreich's PRGs. Specifically, when $d(n) = d$ is constant, Goldreich's PRGs belong to the complexity class NC^0. The constant locality ensures that the PRG can be efficiently implemented in parallel, making it highly scalable across multiple computing cores. This property has attracted significant attention to Goldreich's PRGs.

Cryan and Miltersen [13] first considered the existence of PRGs in NC^0 and obtained a negative result: any PRG in NC^0_3 cannot resist statistical linear tests for $m \geq 4n$. Mossel et al. [22] further ruled out PRGs in NC^0_4 for $m \geq 24n$ but proved the existence of PRGs in NC^0_5. Specifically, they proposed a candidate PRG instantiated with the $\mathsf{XOR\text{-}AND}$ predicate, defined as $P_5(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$, achieving superlinear stretch.

A natural and ambitious goal is to achieve *polynomial stretch*, i.e., $m = n^s$ for some $s > 1$. Goldreich's PRGs with polynomial stretch have garnered special attention due to their wide range of cryptographic applications, such as secure computation with constant computational overhead [18,5,6], indistinguishability obfuscation (iO) [8,9,24,25,19], MPC-friendly primitives [21,17,1], and cryptographic capsules [10]. Applebaum [2] first proved the existence of weak PRGs with polynomial stretch $m = n^s$ for some $s > 1$, having a distinguishing gap of at most $1/n^s$. This result was later improved to $m = n^s$ for $s < 1.25$ using non-degenerate predicates [4], and further extended to $s < 1.5$ for the special case of P_5 [23]. Subsequently, the general polynomial regime $m = n^s$ was considered, and the $\mathsf{XOR\text{-}THR}$ predicate was proposed as a candidate with constant locality $d = O(1)$ [7], where the challenge of $\mathsf{XOR\text{-}MAJ}_{a,b}$ with $a \geq 5s$ and $b > 36s$ was proposed to achieve s-pseudorandomness with proved resistance to existent attacks.

Since the inception of Goldreich's PRGs [16], cryptanalysis has been an integral part of the research on predicate design and PRG construction in NC^0. Early works 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 attacks [12] and improved methods [27] have been proposed, with complexities $O(2^{\frac{1}{2}n^{2-s}})$ and $O(2^{\frac{73}{288}n^{2-s}})$, respectively. The guess-and-decode method [27] further improved experimental results. For the $\mathsf{XOR\text{-}MAJ}$ predicate, a guess-and-determine attack succeeds with complexity $O(n^{\omega}2^{n^{\frac{b-s}{b-1}}})$ [12]. As stated in [11], these attacks are specifically targeted at predicates with very small locality (from 5 to 8), and their complexity becomes prohibitive for predicates with larger locality. Therefore, whether efficient attacks exist for arbitrary XOR-MAJ predicates or even for arbitrary predicate remains an open question.

1.1 Our contributions

In this paper, we give a new attack on Goldreich's PRGs, specifically focused on those instantiated on the XOR-THR predicate. We first show the bit-fixing correlation property of the XOR-THR predicate—that is, by fixing r bits in the THR predicates to 1s, the correlation of the output with a corresponding linear expression increases significantly. Thus, each output bit corresponds to a correlation equation. When the correlation is high, the system can be solved efficiently.

Combining the bit-fixing property and solvers for correlation equations, we give an attack on the XOR-THR predicate. Then we apply the attack to the XOR-MAJ challenges proposed in [3]. The results show that some PRGs with the XOR-MAJ predicate can be inverted efficiently even when the stretch $s = 1$, such that one-wayness cannot be guaranteed. Next, we extend the attacks to an instance used for constructing silent OT [11]. By choosing appropriate parameters, a PRG with a 256-bit secret seed can be broken practically with a cost of around 2^{29} calls to Gaussian elimination. We implement this attack on a cluster of 32 CPU cores and the attack succeeds in around 5.5 hours. Furthermore, we provide some results on so-called *piecewise symmetric* cases, where the predicate can be expressed as a direct sum of some symmetric predicates, in which each one does not rely on the order of the input, and we show that the attack is feasible. All the above results extend the security analysis of Goldreich's PRGs—$\Omega(s)$-resilience and algebraic immunity are insufficient to achieve pseudorandomness or one-wayness with n-bit security guarantee.

Finally, we give key recovery attacks on FiLIP cipher[15], of which some instances are based on the THR related predicates. Our attacks are efficient for most instances. For example, FiLIP with the XOR-THR predicate, whose key size is 1944, can be broken by 2^{95} calls to Gaussian elimination. FiLIP with the XOR-THR-THR predicate, whose key size is 913, can be broken by fewer than 2^{100} calls to Gaussian elimination. In both examples, the assumed security guarantee is $\lambda = 128$.

The rest of the paper is organized as follows. In Section 2, some basic preliminaries will be introduced. Then we show the bit-fixing correlation attack on the XOR-THR predicate in Section 3. Next, we apply our attack to XOR-MAJ based encryption schemes, including the attack on the XOR-MAJ challenges (STOC 2016) and an attack on an instance used for silent OTs (EUROCRYPT 2024), followed by some discussion on general piecewise symmetric predicates in Section 4. In Section 5, we present key recovery attacks on the FiLIP cipher instantiated on THR-related predicates. Finally, we conclude this paper in Section 6.

2 Preliminaries

In this paper, all operations are over the finite field $\mathbb{F}_2$, where addition corresponds to the XOR operation. We denote sets using braces $\{\}$, vectors using parentheses $()$, and use square brackets $[\ ]$ to indicate indices of a vector.

2.1 Goldreich's Pseudorandom Generator

A pseudorandom generator (PRG) maps a short secret random input $x \in \{0,1\}^n$ to a longer pseudorandom output $y \in \{0,1\}^m$. In this paper, we focus on the polynomial regime, where the output length m is a polynomial function of the input length n; specifically, we assume $m = n^s$ for some stretch factor $s \geq 1$.

Goldreich's pseudorandom generator (PRG) employs a fixed predicate P and computes each output bit y_i by applying P to a subset of the input bits. Specifically, to generate each y_i, the generator selects a set σ_i of d distinct indices and computes

$$y_i = P\left(x[\sigma_i]\right),$$

where $x[\sigma_i]$ denotes the vector of input bits at indices specified by σ_i. For convenience, we write $\sum x[\sigma_i] = \sum_{j\in\sigma_i} x_j$, and we use $x[\sigma_i] = 1$ to indicate that $x_j = 1$ for all $j \in \sigma_i$.

A special class of predicates, namely picewise-symmetric predicates, which can be decomposed into k symmetric sub-predicates over disjoint subsets combined through the XOR operation, namely *piecewise symmetric*, is thoroughly explored in this work.

Definition 1 (Symmetric Predicate). *A predicate (or sub-predicate) P over variables $x_1, x_2, \ldots, x_b$ is termed as* symmetric *if, for any permutation $Perm(\cdot)$, it holds that*

$$P(x_1, x_2, \ldots, x_b) = P\big(Perm(x_1, x_2, \ldots, x_b)\big).$$

Accordingly, each output bit y_i can be expressed as

$$y_i = P\big(x[\sigma_i]\big) = P_1\big(x[\sigma_{i,1}]\big) + P_2\big(x[\sigma_{i,2}]\big) + \cdots + P_k\big(x[\sigma_{i,k}]\big),$$

i.e., the j-th sub-predicate P_j operates on the subset $x[\sigma_{i,j}]$ of the input bits to generate one bit output, where $\sigma_{i,j}$ are disjoint and $\sigma_i = \cup_{j=1}^{k}\sigma_{i,j}$. All sub-predicates are symmetric, allowing us to treat their inputs as sets rather than ordered sequences. We assume that there is at least one nonlinear sub-predicate and at least one linear sub-predicate among the P_j. For example, XOR-AND predicate, contains an XOR sub-predicate and an AND predicate is one in the family.

In the typical attack scenario, the predicate P, the input lengths, the index sets $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$, and the outputs y are all accessible to the attacker.

2.2 Predicates

A commonly used linear sub-predicate is the XOR predicate of input x, defined as

$$\mathsf{XOR}_d(x[\sigma]) = \sum x[\sigma],$$

A commonly used nonlinear sub-predicate is the $\mathsf{threshold}$ predicate of input x, defined as

$$\mathsf{THR}_{t,d}(x[\sigma]) = \begin{cases} 1, & \text{if Hamming}(x[\sigma]) \geq t; \\ 0, & \text{otherwise,} \end{cases}$$

where $\text{Hamming}(x[\sigma_i])$ denotes the Hamming weight (i.e., the number of ones) in the vector $x[\sigma_i]$. Majority predicate MAJ is a special case of the threshold predicate and is defined as

$$\mathsf{MAJ}_d(x[\sigma]) = \mathsf{THR}_{\frac{d+1}{2},d}(x[\sigma]),$$

for odd d.

XOR-$\mathsf{Threshold}$ *Predicate:* The XOR-Threshold predicate is a direct sum of an XOR predicate and a $\mathsf{threshold}$ predicate, defined as

$$\mathsf{XOR}_a\text{-}\mathsf{THR}_{t,b}(x[\sigma] = (x[\sigma_1], x[\sigma_2])) = \mathsf{XOR}_a(x[\sigma_1]) + \mathsf{THR}_{t,b}(x[\sigma_2]),$$

where σ_1 and σ_2 are disjoint and $\sigma = \sigma_1 \cup \sigma_2$.

Specifically, when b is odd, the XOR-MAJ predicate can be defined as

$$\mathsf{XOR}_a\text{-}\mathsf{MAJ}_b(x[\sigma]) = \mathsf{XOR}_a(x[\sigma_1]) + \mathsf{MAJ}_b(x[\sigma_2]).$$

2.3 Correlation Equations

The correlation between a function $f(x) \in \{0,1\}$ and a linear expression $L(x) \in \{0,1\}$ is defined as the cardinality of the set of x such that $f(x) = L(x)$ over the number of all possible inputs 2^n, expressed as

$$p = \frac{|\{x \in \{0,1\}^n : f(x) = L(x)\}|}{2^n}.$$

The bias of this correlation is $\epsilon = p - \frac{1}{2}$, so the range of ϵ should be $[-\frac{1}{2}, \frac{1}{2}]$.

If $\epsilon < 0$, it indicates that $f(x)$ is more likely equal to $L(x) + 1$ than $L(x)$. In such cases, we can consider the correlation between $f(x)$ and $L(x) + 1$, which will have a positive bias. Therefore, without loss of generality, we focus on cases where $\epsilon \geq 0$.

Given the output $y = f(x)$, the equation $y = L(x)$ is a correlation equation that holds with probability p. As a special case, constant is also a linear expression.

Algorithm 1 System Solving

```
Input: A group of noisy equations $\mathcal{G}$, number of variables $k$, maximum number of
    equations used $e$
1: $S \leftarrow \min\{e, k\}$ randomly chosen equations
2: $X \leftarrow$ solutions of $S$, including enumeration
3: return $X$
```

For example, suppose P_1 is a linear predicate and P_2 is a nonlinear predicate with input length b. Then each output bit y_i can be used to derive a correlation equation of the form

$$y_i = P_1\big(x[\sigma_{i,1}]\big) + 1,$$

which holds with probability

$$p = \frac{|\big\{x \in \{0,1\}^n : P_2\big(x[\sigma_{i,2}]\big) = 1\big\}|}{2^b}.$$

Here, $x[\sigma_{i,1}]$ and $x[\sigma_{i,2}]$ are disjoint subsets of the input bits x, and $\sigma_{i,1}$ and $\sigma_{i,2}$ are the corresponding index sets.

2.4 Solving Correlation Equations

Throughout the paper, we often need to solve correlation equations involving k variables, using at most e equations. Here, each correlation equation holds with certain probability.

When $e \geq k$, selecting k random equations allows us to form a square system, which can be solved using Gaussian elimination. If the system is of full rank, all k variables can be determined, and a single solution will be found; otherwise, some variables may need to be enumerated, leading to more than one solutions.

On the other hand, if $e < k$, we can only use e equations, so at most e variables can be determined directly, and the remaining variables need to be enumerated.

These two cases can be unified by first selecting $\min\{e, k\}$ equations to form a system. We then solve this system using Gaussian elimination to determine some variables and enumerate the values of the remaining variables, obtaining one or more solutions. We refer to this entire process as the System Solving procedure, outlined in Algorithm 1.

Notably, even solutions are obtained, their correctness can not be verified here as the system is noisy. However, it is obvious that if and only if all correlation equations are correct, the solutions containsexactly one correct solution, which is the correct input, or part of it.

3 Bit-fixing Correlation Attack on Goldreich's PRGs with $\mathsf{XOR\text{-}THR}$ Predicate

In this section, we present inverse attacks on Goldreich's pseudorandom generators (PRGs) that use the $\mathsf{XOR\text{-}THR}$ predicate. Our goal is to recover the secret input seed x using all available output data.

3.1 Observation

The MAJ predicate is a balanced Boolean function, meaning it outputs 0 and 1 with equal probability when its inputs are uniformly random. Consequently, the correlation equation $y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + 1$ holds with probability $p = \frac{1}{2}$. Therefore, any system containing such correlation equations cannot be efficiently solved or even distinguished due to the lack of bias.

However, we observe that by fixing some input bits of the MAJ predicate to either all zeros or all ones, we can introduce a bias into its output. This property is referred to as the *bit-fixing correlation.*

To illustrate this observation, consider the following example.

Example 1. Consider the predicate $P = \mathsf{MAJ}_3$ with input indices $\sigma_i = \{1, 2, 3\}$.

Without fixing any bits, the probability that P outputs 1 is

$$\Pr[P(x[\sigma_i]) = 1] = \frac{\sum_{i=2}^{3} \binom{3}{i}}{2^3} = \frac{1}{2},$$

consistent with the well-known result that MAJ is balanced.

If we fix one bit, say $x_1 = 1$, then the probability becomes

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

Thus, the corresponding bias is $\epsilon = \Pr(P[x[\sigma_i]) = 1 \mid x_1 = 1] - \frac{1}{2} = \frac{1}{4}$.

Similarly, if we fix two bits to ones, say $x_1 = x_2 = 1$, then

$$\Pr[P(x[\sigma_i]) = 1 \mid x_1 = x_2 = 1] = 1,$$

and the bias is $\epsilon = \Pr[P(x[\sigma_i]) = 1 \mid x_1 = x_2 = 1] - \frac{1}{2} = \frac{1}{2}$.

Analogous results can be obtained by fixing certain bits to zeros.

This bit-fixing property exists for the general $\mathsf{THR}_{t,b}$ predicate, as shown in Lemma 1.

Lemma 1. *Given the threshold predicate $\mathsf{THR}_{t,b}$ over an arbitrary set $x[\sigma]$, by fixing $r \leq t$ bits of indices $\rho = \{\rho_1, \rho_2 \cdots \rho_r\}$ to ones, where $\rho \subset \sigma$, i.e., $x[\rho] = 1$, we have*

$$p = \Pr[\mathsf{THR}_{t,b}(x[\sigma]) = 1 | x[\rho] = 1] = \frac{1}{2^{b-r}} \sum_{t-r}^{b-r} \binom{b-r}{i}. \tag{1}$$

As a special case, when $t = \frac{b+1}{2}$ and b is odd,

$$p = \Pr[\mathsf{MAJ}_b(x[\sigma]) = 1 | x[\rho] = 1] = \frac{1}{2^{b-r}} \sum_{i=(b+1)/2-t}^{b-r} \binom{b-r}{i}, \tag{2}$$

and the corresponding bias

$$\epsilon = p - \frac{1}{2} = \frac{1}{2^{b-r+1}} \sum_{i=(b+1)/2-t}^{(b-1)/2} \binom{b-r}{i}. \tag{3}$$

Proof. See Appendix A.

Based on the property, the holding probability of the correlation equation derived from the general XOR-THR predicate can be calculated by Theorem 1.

Theorem 1. *Given the predicate XOR_a-$\mathsf{THR}_{t,b}$, by fixing r indices, ρ_i, in $\sigma_{i,2}$ to 1s, the following correlation equation holds with probability*

$$\Pr[y_i = \mathsf{XOR}_a(x[\sigma_{i,1}])+1] = \Pr[\mathsf{THR}_{t,b}(x[\sigma_{i,2}]) = 1 | x[\rho_i] = 1] = \frac{1}{2^{b-r}} \sum_{i=t}^{b-r} \binom{b-r}{i}. \tag{4}$$

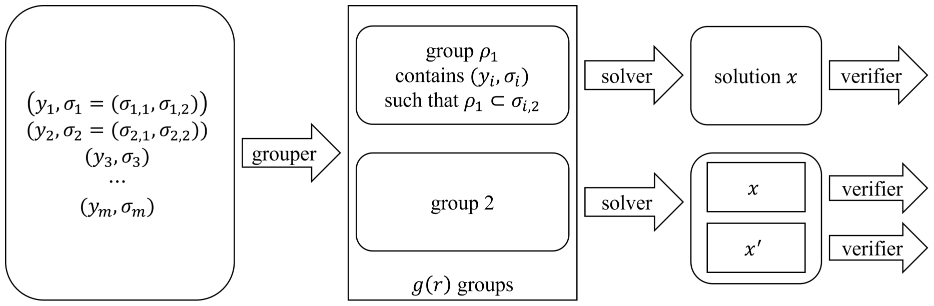
3.2 Bit-fixing Correlation Attack

To recover the secret seed or input x, the simplest method is to find all the correlation equations corresponding to the outputs, randomly select some to form a noisy system, and then solve it using system solving methods. However, since the bias of a randomly selected system is small, according to our discussion in Section2.4, system solving can yield the correct solution only when all equations in the system are correct. Therefore, this random selection is very inefficient. By leveraging the *bit-fixing property*, if some input bits of certain indices can be fixed to 1s or 0s, the bias of the correlation equations becomes very high.

Thus, the *bit-fixing correlation attack* naturally divides into two phases, as shown in Figure 1.:

1. The first phase increases the bias by achieving the bit-fixing effect through *grouping* according to sharing bits.
2. The second phase randomly selects equations to construct noisy systems and finds the correct solution through *system solving* and verification.

Grouping Phase In this phase, we aim to collect enough correlation equations by exploiting the bit-fixing property of the THR predicate. Although the values of the input bits are unknown, we can group the equations based on common indices in their THR inputs $\sigma_{i,2}$. Let ρ_j denote the set of common indices for



$(y_1, \sigma_1 = (\sigma_{1,1}, \sigma_{1,2}))$
$(y_2, \sigma_2 = (\sigma_{2,1}, \sigma_{2,2}))$
(y_3, σ_3)
$\cdots$
(y_m, σ_m)
grouper
group ρ_1 contains (y_i, σ_i) such that $\rho_1 \subset \sigma_{i,2}$
group 2
$g(r)$ groups
solver
solver
solution x
x
x'
verifier
verifier
verifier

Fig. 1. Framework of Bit-fixing Correlation Attack

Algorithm 2 Equation Grouping

```
Input: Number of fixing bits $r$, number of groups $g$
 1: for $j = 1$ to $g$ do
 2:     $\rho_j \leftarrow$ randomly chosen a set of $r$ distinct indices, each $\in [1, n]$
 3:     $\mathcal{G}_j \leftarrow \emptyset$
 4:     for $i = 1$ to $m$ do
 5:         if $\rho_j \subset \sigma_{i,2}$ then
 6:             $\mathcal{G}_j$.append($y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + 1$)
 7:         end if
 8:     end for
 9: end for
10: return $(\mathcal{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_g)$ and $(\rho_1, \rho_2 \cdots \rho_g)$
```

group $\mathcal{G}_j$. If the bits corresponding to ρ_j are all ones, the bias of the correlation equations in $\mathcal{G}_j$ becomes significant. By creating a sufficiently large number of groups (e.g., 2^r), there is an non-negligible probability that at least one group will have all its common bits equal to ones.

Given an integer r and a number of groups g, the grouping phase is described in Algorithm 2. First, we randomly generate g different indices sets $(\rho_1, \rho_2 \cdots \rho_g)$, each containing r distinct indices, and initialize the corresponding empty groups $\mathcal{G}_j$. Then, for each $\sigma_i \in \Sigma$, we append the corresponding equation $y_i = \mathsf{XOR}_a(x[\sigma_{i,1}]) + 1$ to group $\mathcal{G}_j$ if and only if $\rho_j \subseteq \sigma_{i,2}$.

Thus, the time complexity of Algorithm 2 is

$$T_{\text{group}} = mg. \tag{5}$$

Next, we analyze the expected number of equations in each group. Each correlation equation can be appended to $\binom{b}{r}$ groups, since there are $\binom{b}{r}$ ways to choose r indices from the b bits in $\sigma_{i,2}$. There are at most $\binom{n}{r}$ possible groups. Therefore, the expected number of equations in each group is

$$e = m\frac{\binom{b}{r}}{\binom{n}{r}} = \frac{n^s\binom{b}{r}}{\binom{n}{r}}. \tag{6}$$

Algorithm 3 Attack with Gauss

```
Input: Number of fixed bits $r$
 1: $e \leftarrow \frac{n^s\binom{b}{r}}{\binom{n}{r}}$, $p \leftarrow \frac{1}{2^{b-r}}\sum_{i=t-r}^{b-r}\binom{b-r}{i}$, $g \leftarrow \frac{2^r}{p^{\min\{e,n-r\}}}$
 2: Grouping Phase:
 3: $(\mathcal{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_g), (\rho_1, \rho_2 \cdots \rho_g) \leftarrow$ Equation Grouping $(r, g)$
 4: Solving Phase:
 5: for $j = 1$ to $g$ do
 6:     $X_j \leftarrow$ System Solving $(\mathcal{G}_j, n-r, e)$
 7:     for $x \in X_j$ do
 8:         if $P(x[\sigma_i]|x[\rho_i] = 1) = y_i$ for all $i \in [1, m]$ then
 9:             return $x$
10:         end if
11:     end for
12: end for
13: return $\perp$
```

Solving Phase After the grouping phase, while the total number of correct equations remains the same, the grouping may result in certain groups containing significantly more correct equations. Therefore, our next goal is to find a group that contains equations with a high bias, solve the equations in that group, and then verify the solution. We aim to efficiently execute this process, which we call the *solving phase*.

In this subsection, we introduce two solving algorithms inspired by [14], namely **Gauss** and **SubGauss**. Since these solvers are straightforward, we present them along with the entire attack process.

Gauss Solver: The Gauss solver is a direct extension of Gaussian elimination. The combined grouping and solving process using the Gauss solver is outlined in Algorithm 3.

First, we choose the parameter r, which will be determined by the complexity analysis in Section3.3. We compute the probability p using Equ. (1) and the expected number of equations per group e using Equation (6). We then set

$$g = \frac{2^r}{p^{\min\{n-r,e\}}} \tag{7}$$

In the *grouping phase*, using (r, g) as input, we execute Algorithm 2 to obtain g groups, each containing approximately e equations.

In the *solving phase*, we process each group using the Gauss solver. For each group $\mathcal{G}_j$, we perform Gaussian elimination and, if necessary, enumerate the remaining variables to obtain candidate solutions, denoted as X_j. Each candidate solution $x \in X_j$ is verified by checking whether $P(x[\sigma_i]|x[\rho_i] = 1) = y_i$ holds for all $i \in [1, m]$.

SubGauss Solver: SubGauss is still based on Gaussian elimination, but uses fewer groups with each containing more equations. Concretely, $g = 2^r$ and the required number of equations in each group is $(n - r)/p$. Compute p by Equ. (1) and e by Equ. (6).

Algorithm 4 Attack with SubGauss

```
Input: Number of fixing bits $r$
1: $e \leftarrow \frac{n^s\binom{b}{r}}{\binom{n}{r}}$,$p \leftarrow \frac{1}{2^{b-r}}\sum_{i=t-r}^{b-r}\binom{b-r}{i}$,$g \leftarrow 2^r$
2: Grouping Phase:
3: $(\mathcal{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_g), (\rho_1, \rho_2 \cdots \rho_g) \leftarrow$ Equation Grouping $(r, g)$
4: Solving Phase
5: for $j = 1$ to $g$ do
6:     for $i = 1$ to $\frac{1}{p^{\min\{ep, n-r\}}}$ do
7:         $X_j \leftarrow$ System Solving $(\mathcal{G}_j, n-r, ep)$
8:         for $x \in X_j$ do
9:             if $P(x[\sigma_i]|x[\rho_i] = 1) = y_i$ for all $i \in [1, m]$ then
10:                return $x$
11:            end if
12:        end for
13:    end for
14: end for
15: return $\perp$
```

As shown in Algorithm 4, in the *grouping phase*, `Equation Grouping` (Algorithm 2) is performed, such that g groups with 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, there are two cases. When $e > (n-r)/p$, in a group, perform Gaussian elimination on randomly chosen $n-r$ equations and verify the solutions, which is executed $1/p^{n-r}$ times for each group . When $e < (n-r)/p$, guess $n-r-ep$ bits, substitute and perform Gaussian elimination on randomly chosen ep equations and verify the solutions, which is executed $1/p^{ep}$ times for guess and each group.

3.3 Analysis of Complexity and Success Ratio

In this section, we present the complexity and success ratio analysis.

Attack with Gauss Revisiting Algorithm 3, the complexity of the solving phase is

$$\begin{cases} g(T_g + T_v), & e \geq n-r, \\ g2^{n-r-e}(T_g + T_v), & e < n-r, \end{cases}$$

where T_g and T_v denotes the cost of Gaussian elimination and verification, respectively. For the ease of analysis, we use n^ω to represent $T_g + T_v$, where ω is a constant.

As the complexity of *grouping phase* is $n^s g$, by substituting g by Equ. (7) the total complexity of the attack with Gauss solver is

$$T_{Gauss} = \begin{cases} \frac{2^r}{p^{n-r}}(n^s + n^\omega), & e \geq n-r, \\ \frac{2^r}{p^e}(n^s + 2^{n-r-e}n^\omega), & e < n-r. \end{cases} \quad (8)$$

Now we analyze the success ratio of the attack with Gauss solver. The attack succeeds when 1) r-bit setting is correct (r bits are ones) and 2) the solution obtained from at least 1 group is correct. Each out of g groups satisfies the above two constraints simultaneously with the probability $\frac{1}{g}$, hence the success ratios of the attack with Gauss solver is

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

where E denotes the base of natural logarithm.

In order to guarantee that there are enough groups, the following inequality holds

$$\binom{n}{r} \geq 2^r \left(\frac{1}{p}\right)^{\min(e,n-r)}.$$

Increasing r increases p but decreases e. When $e > n - r$, it is more efficient to increase r. But when $e < n - r$, the complexity of *solving phase* may increase significantly when increasing r. The results are confirmed by the attacks in Section4. The attack with Gauss solver can be reduced to minimize T_{Gauss} (Equ. (8)) by choosing appropriate r with success ratio $\frac{E-1}{E}$ and constraint $\binom{n}{r} \geq \frac{2^r}{p^{\min(e,n-r)}}$, where p and e are computed by Equ. (1) and Equ. (6), respectively.

Attack with SubGauss Revisiting Algorithm 4, the complexity of the *solving phase* is

$$\begin{cases} g\frac{1}{p^{n-r}}n^{\omega}, & e \geq (n-r)/p, \\ g2^{n-r-ep}\frac{1}{p^{ep}}n^{\omega}, & e < (n-r)/p, \end{cases}$$

where n^{ω} term denotes the cost of Gaussian elimination and verification.

As the complexity of *grouping phase* is $n^s g$, by substituting $g = 2^r$, the total complexity of attack with SubGauss solver is

$$T_{SubGauss} = \begin{cases} 2^r(n^s + \frac{1}{p^{n-r}}n^{\omega}), & e \geq (n-r)/p, \\ 2^r(n^s + \frac{2^{n-r-ep}}{p^{ep}}n^{\omega}), & e < (n-r)/p. \end{cases} \tag{9}$$

In order to obtain a right guess of the r bits, enough sets are needed with the rough computing

$$\binom{n}{r} \geq 2^r.$$

Now we analyze the success ratio of the attack with SubGauss solver. The attack instantiated on SubGauss algorithm succeeds when 1) the r-bit setting is correct (r bits are ones) and 2) in the correct r-bit setting, at least 1 choice of ep equations are correct. The former holds with probability $1 - (1 - \frac{1}{2^r})^{2^r} = 1 - \frac{1}{E}$. For the latter case, let $e' = \min(e, (n-r)/p)$, each out of $1/p^{e'}$ 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 $1 - \left(1 - p^{e'p}\right)^{1/p^{e'p}} = 1 - \frac{1}{E}$.

Summed up, the success ratio of the attack with SubGauss solver is

$$p_{sucSubGauss} = (1 - \frac{1}{E})^2 = \frac{(E-1)^2}{E^2},$$

where E denote the base of natural logarithm.

Similarly, increasing r can lead to larger p but decrease e. If e is large enough, it is more efficient to increase r. But when $e < (n-r)/p$, the complexity increases significantly due to smaller e. The results are confirmed by the attacks in Section 4. The attack with Gauss solver can be reduced to minimize $T_{subGauss}$ (Equ. (9)) by choosing appropriate r with success ratio $\frac{(E-1)^2}{E^2}$ and constraint $\binom{n}{r} \geq 2^r$, where p and e are computed by Equ. (1) and Equ. (6), respectively.

Further Discussion In Section 3.2, we provide two types of solvers to address the linear noisy system of correlation equations. They perform well in different cases, discussed as follows.

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

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

As $p > 1/2$, T_{Gauss} over $T_{SubGauss}$ is usually smaller than 1, hence Gauss solvers outperforms 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 on SubGauss solver is obviously more efficient than that on Gauss solver as they share the same solving complexity but the grouping complexity of SubGauss is reduced by a factor of $\frac{1}{p^{n-r}}$.

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

4 Attacks on XOR-THR Based Schemes

In this section, we present attacks on schemes that utilize the picewise-symmetric predicate. First, we demonstrate an attack on the XOR-MAJ challenges proposed at STOC 2016, followed by an attack on an instance used for constructing silent Oblivious Transfers (OTs).

4.1 Attack on the Challenges at STOC 16

Table 1. The parameters for the attack on the challenge with $a = 5s$ and $b = 37s$ for different stretches s. In order to guarantee valid Majority predicate, b is set to the least odd integer for $b \geq 37s$. The output length is $m = n^s$. For Gauss solver and SubGauss solver, corresponding parameters are chosen accordingly. p denotes the probability each equations holds. e denotes the expected number of equation in each group and ep denotes 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.

s	(n, a, b)	Solver	r	p	e	ep	T_{eg}	T_s
1	(42,5,37)	Gauss	5	0.8115	21	17	2^{17}	$\mathbf{2^{28}}$
		SubGauss	7	0.8998	16	14	2^{13}	$\mathbf{2^{30}}$
	(80,5,37)	Gauss	1	0.5660	37	20	2^{38}	$\mathbf{2^{74}}$
		SubGauss	1	0.5660	37	20	2^{8}	$\mathbf{2^{77}}$
	(128,5,37)	Gauss	1	0.5660	37	20	2^{39}	$\mathbf{2^{122}}$
		SubGauss	1	0.5660	37	20	2^{8}	$\mathbf{2^{125}}$
1.25	(54,7,47)	Gauss	9	0.9283	37	34	2^{20}	$\mathbf{2^{21}}$
		SubGauss	7	0.8659	51	44	2^{14}	$\mathbf{2^{19}}$
	(80,7,47)	Gauss	3	0.6742	47	31	2^{38}	$\mathbf{2^{60}}$
		SubGauss	2	0.6170	81	50	2^{11}	$\mathbf{2^{65}}$
	(128,7,47)	Gauss	2	0.6170	57	35	2^{51}	$\mathbf{2^{111}}$
		SubGauss	1	0.5585	157	88	2^{10}	$\mathbf{2^{114}}$
1.5	(65,8,57)	Gauss	12	0.9638	92	88	$\mathbf{2^{24}}$	2^{15}
		SubGauss	9	0.9033	147	133	$\mathbf{2^{18}}$	2^{17}
	(80,8,57)	Gauss	7	0.8389	59	49	2^{32}	$\mathbf{2^{36}}$
		SubGauss	6	0.7995	86	69	2^{15}	$\mathbf{2^{33}}$
	(128,8,57)	Gauss	3	0.6583	119	78	$\mathbf{2^{85}}$	2^{81}
		SubGauss	2	0.6061	274	166	2^{12}	$\mathbf{2^{93}}$

At STOC 2016, the authors introduced challenges instantiated on the XOR-MAJ predicate [3] for cryptanalysis, specifically,

$$\mathsf{XOR}_a\text{-}\mathsf{MAJ}_b, \tag{10}$$

where $d = a + b$ and $m = n^s$. It was claimed that the challenge (10) is secure when $a \geq 5s$ and $b > 36s$.

Here, we provide results for different stretches s and key sizes n. Since our attack is independent of a, we fix $a = 5s$ for our analysis. The approximate parameter r, representing the number of fixed bits, can be determined experimentally. The corresponding probability p and the expected number of equations in each group e are computed using Equ. (1) and Equ. (6), respectively. The results are summarized in Table 1.

Note that in the cases highlighted in red, the expected number of equations exceeds the required number. For example, in the case where $(n, a, b) = (65, 8, 57)$, it is required that $(e, ep) = (71, 64)$. Therefore, we use $71/157$ of the total $m = n^s = 65^{1.5} \approx 524$ bits. The complexity of the grouping phase can be reduced by a factor of $71/157$. However, since the grouping phase costs much less than the solving phase for the parameter $r = 9$ with the SubGauss solver, this optimization can be neglected. A similar observation applies to the case $(n, a, b) = (128, 8, 57)$.

We demonstrate that fixing just a few bits can achieve highly biased correlation equations. Taking the example with $(n, a, b) = (65, 8, 57)$, where the threshold of the majority predicate is 29, by fixing 9 bits to ones, the probability that each equation holds can reach as high as 0.9033. This instance can be broken with approximately 2^{17} calls to Gaussian elimination and substitution of solutions for verification.

When the stretch is small (e.g, $s = 1$) and the output is short, the attack with Gauss solver outperforms SubGauss solver. However, when the output is relatively long, ($s = 1.5$ for $n = 65$), SubGauss solvers beats Gauss solver. This is because Gauss solver needs fewer equations in each group, confirming our analysis in Section 3.3.

4.2 Attack on the Instance at EUROCRYPT24

The XOR_a-$\mathsf{THR}_{t,b}$ predicate was used for constructing fast silent OT with the parameter $(n, m, b, t) = (256, 2^{40}, 64, 32)$ [11]. In this section, we show how to break this instance and implement the attack on PCs.

For $\mathsf{THR}_{32,64}$, by heuristic of r, we obtain the probability p that each correlation equation holds according to Equ. (2), compute e according to Equ. (6) and then obtain ep. Then the complexities for Gauss solver and SubGauss solver are computed by Equ. (8) and Equ. (9), respectively. We show the complexities of grouping phase and solving phase in separate columns for clear presentation. The results are shown in Table 2.

Experiment By Table 2, $r = 10$ is optimal for trading off the grouping complexity T_{eg} and the solving complexity T_s. In practice, Gaussian elimination is much more expensive than each operation in the *grouping phase*, hence we choose $r = 11$ for implementation. Using $2^{31.0}$ output bits, the attack can be realized with $2^{42.0}$ costs for the grouping phase and $2^{28.9}$ calls to Gaussian elimination for the solving phase, as shown in Table 2.

The attack was utilized on two PCs, each with 16 i7-13700 CPU cores. Taken the SHA-256 hash of the string "EUROCRYPTO2025" as the seed, we generated

Table 2. The parameters for breaking the instance [11]. 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.

r	p	solver	D	e	ep	T_{eg}	T_s
7	0.8554	Gauss	$2^{22.3}$	249	212	$2^{85.4}$	$2^{63.1}$
		SubGauss	$2^{22.6}$	291	249	$2^{29.6}$	$2^{63.1}$
8	0.8856	Gauss	$2^{24.5}$	248	219	$2^{75.9}$	$2^{51.5}$
		SubGauss	$2^{24.6}$	280	248	$2^{32.6}$	$2^{51.5}$
9	0.9115	Gauss	$2^{26.6}$	247	225	$2^{68.6}$	$2^{42.0}$
		SubGauss	$2^{26.7}$	270	247	$2^{35.7}$	$2^{42.0}$
10	0.9332	Gauss	$2^{28.8}$	246	229	$2^{63.3}$	$2^{34.6}$
		SubGauss	$2^{28.9}$	263	246	$2^{38.9}$	$2^{34.6}$
11	0.9508	Gauss	$2^{30.9}$	245	232	$2^{59.8}$	$2^{28.8}$
		SubGauss	$2^{31.0}$	257	245	$2^{42.0}$	$2^{28.9}$
12	0.9648	Gauss	$2^{33.1}$	244	235	$2^{57.8}$	$2^{24.6}$
		SubGauss	$2^{33.2}$	252	244	$2^{45.2}$	$2^{24.6}$
13	0.9756	Gauss	$2^{35.4}$	243	237	$2^{57.0}$	$2^{21.7}$
		SubGauss	$2^{35.4}$	249	243	$2^{48.4}$	$2^{21.7}$
14	0.9836	Gauss	$2^{37.6}$	242	238	$2^{57.4}$	$2^{19.8}$
		SubGauss	$2^{37.6}$	246	242	$2^{51.6}$	$2^{19.8}$
15	0.9894	Gauss	$2^{39.9}$	241	238	$2^{58.6}$	$2^{18.7}$
		SubGauss	$2^{39.9}$	243	241	$2^{54.9}$	$2^{18.7}$

2^{31} output bits and store the output bits together with the indices of the input bits in the disk. The SHA-256 function and pseudorandom generator in the c++ library were applied. The generation process cost around 14 hours on a single CPU core. Then the data was copied to the other PC.

After the output bits were generated, *grouping* and *solving* phase were executed. Concretely, each CPU core of two PCs randomly selected $r = 11$ indices and executed `Equation Grouping` algorithm. Then `Gaussian elimination` was performed on 245 randomly chosen equations and the corresponding solutions were substituted into the encryption scheme for verification, which is repeated $1/0.9508^{245} \approx 233428$ times for each group.

We conducted the experiment twice on the same output bits. The correct seed can be recovered after evaluating around 1600 groups, in each of which performing `Gaussian elimination` on 245 randomly chosen equations was repeated 100000 times, costing 5.5 hours in the cluster of 32 CPU cores. Then we re-executed the attack by repeatedly performing `Gaussian elimination` on 245 randomly chosen equations 233428 times for each group. The secret seed was recovered by evaluating 500 groups, taking 4.7 hours. The implementation code is available at `https://github.com/Analytic233/-Bit-fixing-Correlation-Attack-on-Random-Local-Functions`.

Note that our implementation is less optimized. In this attack, the Hamming weight of each correlation equation is $a = 10$ while there are $n - r = 245$ unknowns, i.e., the equations are sparse. As is widely known, a sparse matrix can be inverted with quadratic complexity [26]. Hence, the attack can be optimized furthermore and is feasible on a single PC in practice.

4.3 Generation to Piecewise Symmetric Predicates

In this section, we generalize the bit-fixing correlation attack on Goldreich's pseudorandom generators (PRGs) to more general $\mathsf{XOR}\text{-}X$ predicates, where X is the sum of multiple symmetric sub-predicates. While the general case involves multiple sub-predicates, we focus on the scenario with a single sub-predicate, as other cases can be addressed by combining the $\mathsf{XOR}\text{-}X$ case with the Piling-up Lemma. The similar approach is applied to attacks on the FiLIP ciphers in Section 5.2. Here, we consider the case where the stretch factor s is small.

Similar to the attack on $\mathsf{XOR}\text{-}\mathsf{THR}$ based PRGs, we need to analyze the bit-fixing property and compute the probability that a derived correlation equation holds when fixing r bits to specific values. The bit-fixing correlation attack consists of two phases: the *grouping phase* and the *solving phase*. The grouping phase is identical to Algorithm 2. When s is small, so that $m = n^s$ is small, we consider the Gauss solver to minimize the number of equations required in each group. The attack is identical to Algorithm 3 with the exception of p, which is replaced by that for arbitrary case.

Assuming there are e correlation equations in each group, and each equation holds with probability p, the probability that all equations in a group are correct is p^e. By selecting $1/p^e$ such systems, we can obtain, on average, a linear system consisting entirely of correct equations. As discussed previously, the total complexity is

$$T_{sym} = \begin{cases} \frac{2^r}{p^{n-r}}(n^s + n^\omega), & e \geq n - r, \\ \frac{2^r}{p^e}(n^s + 2^{n-r-e} n^\omega), & e < n - r. \end{cases} \tag{11}$$

Consider a b-arity symmetric predicate. By fixing r bits to specific values (e.g., ones for the THR predicate), the optimal correlation is no smaller than $\frac{1}{2} + 2^{\frac{r-b}{2}-1}$.

Further Discussion Now, we consider a special case that $b = cn^{1/s} + s$, where $c \geq 1$.

For the bit-fixing correlation attack, we can set $r = s$. By Equ. (6), we have $e = \frac{n^s \binom{b}{s}}{\binom{n}{s}} \geq n > n - r$. The complexity, by omitting the n^s term and n^ω term, is then upper-bounded by

$$T_{\text{sym}} = \left(\frac{1}{\frac{1}{2} + 2^{\frac{r-b}{2}-1}} \right)^{n-s} 2^s = \left(\frac{2}{1 + 2^{\frac{r-b}{2}}} \right)^{n-s} 2^s = 2^{(n-s)(1-\log_2 E \cdot 2^{-0.5cn^{1/s}})+s}. \tag{12}$$

Take $(n, s, b) = (512, 3, 19)$ as an example. By Equ. (12), the PRGs with 19-ary arbitrary symmetric predicate can be inverted by 2^{509} calls to Gaussian elimination.

Revisiting the $\mathsf{XOR\text{-}MAJ}$ predicate, one in the piecewise symmetric family, by fixing $r \ll b$ bits to zeros or ones, the correlation is $\frac{1}{2} + \frac{r}{\sqrt{2\pi b}}$, as justified in Corollary 1.

Corollary 1. *Given a majority function $f(x)$ over variables $x_1, x_2, \ldots, x_b$, when $r \ll b$ bits are fixed to a constant value δ, i.e., $x_{i_1} = x_{i_2} = \cdots = x_{i_r} = \delta$, the correlation is approximately*

$$p \approx \frac{1}{2} + \frac{r}{\sqrt{2\pi b}}. \tag{13}$$

Corollary 1 can be proved by combining Lemma 1 and Stirling's approximation. The complete proof is presented in Appendix B.

Consider the same case that $b = cn^{1/s} + s$, where $c \geq 1$. Substituting Equ. 13, omitting the n^s and n^ω terms, the complexity is upperly bounded by

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

Taking the same example $(n, s, b) = (512, 3, 19)$, the complexity is around 2^{191}, which is much lower than the general piecewise symmetric case, indicating that $\mathsf{XOR\text{-}THR}$ is far from well designed predicate to resist bit-fixing correlation attack.

5 A Key-recovery Attack on FiLIP

FiLIP [20] is an improved version of the FLIP cipher [21] to achieve higher security and better MPC performance (in terms of throughput and latency) and FHE performance (in terms of noise). Compared with the instances in [20], the instances in [15] adopt smaller key sizes with fewer output bits with extensive cryptanalysis, thereby reducing computation and storage requirements. Our attacks focus on these instances. In [15], the authors presented two types of FiLIP ciphers based on the THR related predicates, namely, instances with the $\mathsf{XOR\text{-}THR}$ predicate and the $\mathsf{XOR\text{-}THR\text{-}THR}$ predicate. We provide attacks on these instances in Sections 5.1 and 5.2, respectively, given the output y_i $(i = 1, \ldots, m)$ with the indices σ_i and noise w_i known.

Given a secret key x of length n, the keystream y is generated similarly to the PRGs discussed earlier, with the key difference being that the input to the predicate P is the sum (over $\mathbb{F}_2$) of the key bits and a publicly known noise vector w_i. As is shown in Fig. 2, each output bit y_i of FiLIP is generated by

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

where w_i is a vector of publicly known noise bits.

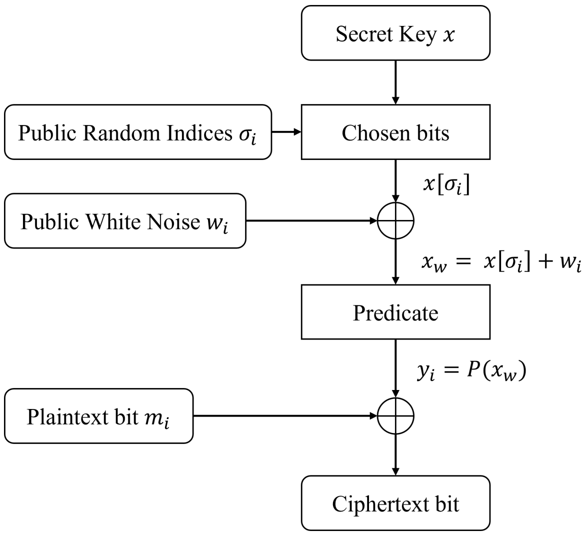


Fig. 2. Encryption Scheme of FiLIP

5.1 Attack on FiLIP Instantiated on 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, and additions are over $\mathbb{F}_2$.

Due to the whitening (i.e., addition of noise bits), the inputs to the predicate may differ even when the indices are the same. Therefore, the original grouping phase cannot be directly applied in the attack, as it assumes that the same indices always lead to the same predicate input values.

To address this issue, we introduce a new grouping algorithm called *data filtering*:

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 $r \leq |\sigma_{i,2} \cap \mathcal{L}| \leq l$.

Next, we attempt to guess the values of x at the indices in $\mathcal{L}$, i.e., $x[\mathcal{L}]$. For each guess, we initialize an empty group $\mathcal{G}$. For each tuple (y_i, σ_i) in $\mathcal{B}$, we consider the corresponding correlation 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

$$x[\mathcal{R}_i] + w_i[\mathrm{pos}(\mathcal{R}_i, \sigma_i)] = 1. \tag{14}$$

Algorithm 5 Attack on FiLIP Instantiated on $\mathsf{XOR}_a\text{-}\mathsf{THR}_{t,b}$

```
Input: Number of fixing bits $r$, larger range $l$.
 1: $e \leftarrow \sum_{u=r}^{l} \sum_{v=r}^{u} mp_u 2^{-u} \binom{u}{v}, p \leftarrow \frac{1}{2^{b-r}} \sum_{i=t-r}^{b-r} \binom{b-r}{i}$
 2: Data Filtering Phase:
 3: $\mathcal{B} \leftarrow \emptyset$, randomly chosen $l$ indices forming $\mathcal{L}$
 4: for $i = 1$ to $m$ do
 5:     if $r \leq |\sigma_{i,2} \cap \mathcal{L}| \leq l$ then
 6:         $\mathcal{B}$.append $(y_i, \sigma_i)$
 7:     end if
 8: end for
 9: Guess and Solve Phase:
10: for a guess $x[\mathcal{L}] \in \{0,1\}^l$ do
11:     $\mathcal{G} \leftarrow \emptyset$
12:     for $(y_i, \sigma_i) \in \mathcal{B}$ do
13:         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
14:             $\mathcal{G}$.append $(y_i = \mathsf{XOR}(x[\sigma_{i,1}] + w_i[1:a]) + 1)$
15:         end if
16:     end for
17:     $X \leftarrow$ System Solving $(\mathcal{G}, n-r, ep)$
18:     for $x \in X$ do
19:         if $P(x[\sigma_i] + w_i | x[\mathcal{L}]) = y_i$ for all $i \in [1, m]$ then
20:             return $x$
21:         end if
22:     end for
23: end for
24: return $\perp$
```

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 $x_j + w_i[\mathrm{pos}(j, \sigma_i)] = 1$.

After grouping, we apply the SubGauss solver to each group $\mathcal{G}$ and verify the solutions by substituting them back into the FiLIP cipher. The attack is detailed in Algorithm 5.

Analysis of Complexity and Success Ratio 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

$$p_u = \frac{\binom{l}{u}\binom{n-l}{b-u}}{\binom{n}{b}}. \tag{15}$$

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

It is cleat that the complexity of *guess and solve* step is $2^l(\sum_{u=r}^{l} mp_u + T_s)$, where T_s denotes the complexity of solving a group of correlation equations and verifying solutions. In order to evaluate T_s, we first evaluate the expected number of correlation 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 $2^{-u} \sum_{v=r}^{u} \binom{u}{v}$. Hence, the expected number of correlation equations in each group is

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

Similar to the analysis in Section 3.3, the complexity of the attack on FiLIP instantiated on XOR-THR predicate is

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

Hence, the total complexity is

$$T_{xt} = \begin{cases} 2^l m \sum_{u=r}^{l} p_u + \dfrac{2^l}{p^{n-l}} n^{\omega}, & e \geq (n-l)/p, \\ 2^l m \sum_{u=r}^{l} p_u + \dfrac{2^{n-ep}}{p^{ep}} n^{\omega}, & e < (n-l)/p. \end{cases} \tag{17}$$

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 5 succeeds with the probability $1 - \frac{1}{E}$, where E denotes the base of natural logarithm.

Table 3. Attack Parameters for Reduced FiLIP with $\mathsf{XOR_a}$-$\mathsf{THR_{t,b}}$.

a	t	b	n	r	l	p	D	e	ep	T_{eg}	T_s	2^{λ}
40	52	104	530	16	29	0.9653	$2^{39.5}$	518	501	$2^{52.2}$	$2^{54.5}$	2^{80}
100	24*	44	982	10	34	0.9878	$2^{39.8}$	959	948	$2^{53.3}$	$2^{50.7}$	2^{80}
65	32	63	2560	15	60	0.9853	$2^{64.0}$	2537	2500	$2^{85.7}$	$2^{113.2}$	2^{128}
58	35	70	4096	14	68	0.9780	$2^{63.8}$	4118	4028	$2^{93.5}$	$2^{197.4}$	2^{128}
80	260	520	1200	41	56	0.9726	$2^{63.0}$	1176	1144	$2^{100.8}$	$2^{101.9}$	2^{128}
70	93	186	1499	26	60	0.9838	$2^{63.8}$	1462	1439	$2^{93.7}$	$2^{94.0}$	2^{128}
65	96	191	1461	27	63	0.9827	$2^{63.3}$	1422	1398	$2^{97.3}$	$2^{98.3}$	2^{128}
70	61	122	1777	21	61	0.9860	$2^{63.5}$	1740	1716	$2^{91.1}$	$2^{96.0}$	2^{128}
160	48	96	1987	19	64	0.9890	$2^{63.9}$	1944	1923	$2^{92.7}$	$2^{94.7}$	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 5.

The parameters r and l are chosen to minimize the complexity T_{xt} in Equ. (17). We provide by heuristic the results, as shown in Table 3. The results demonstrate the efficiency of our attacks for most instances. For example, FiLIP cipher with XOR_{100}-$\mathsf{THR}_{24,44}$ can be broken by no more than 2^{51} calls to Gaussian elimination, while the key size is 982 and security assumption is $\lambda = 80$.

5.2 Attack on FiLIP Instantiated on XOR-THR-THR

In the same paper [15], another instance with the XOR-THR-THR predicate was proposed, defined as:

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

where $\sigma_{i,j}$ $(j = 1, 2, 3)$ are disjoint and $\sigma_i = \sigma_{i,1} \cup \sigma_{i,2} \cup \sigma_{i,3}$.

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 difference lies in the probability that a correlation equation holds.

For the XOR-THR-THR predicate, 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, a correlation equation can be derived, as justified in Theorem 2.

Theorem 2. *The equation*

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

holds with probability

$$p = \frac{1 + (2p_1 - 1)(2p_2 - 1)}{2},$$

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. Theorem 2 can be proved by combining Lemma 1 and the well-known Piling-up Lemma.

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

First, randomly 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

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

Next, 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

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

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 $a+b_1+b_2$.

After forming the group $\mathcal{G}$, we apply the SubGauss solver and verify the solutions by substituting them back into the FiLIP cipher. The attack is detailed in Algorithm 6.

Table 4. Choosing Equation Attack Parameters for FiLIP Instantiated on $\mathsf{XOR_a}$-$\mathsf{THR_{t_1,b_1}}$-$\mathsf{THR_{t_2,b_2}}$.

a	t_1	b_1	t_2	b_2	n	r_1	r_2	l	p	D	e	ep	T_{eg}	T_s	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^{80}
100	11	22	11	22	397	7	7	40	0.9655	$2^{38.8}$	369	357	$2^{61.1}$	$2^{58.1}$	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^{128}
54	22	45	23	45	3072	9	10	111	0.9255	$2^{63.9}$	3199	2961	$2^{140.1}$	$2^{441.6}$	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}
65	47	95	48	96	830	16	18	66	0.9642	$2^{63.9}$	792	764	$2^{104.5}$	$2^{106.2}$	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^{128}
60	26	53	27	53	1229	11	13	74	0.9598	$2^{63.8}$	1203	1155	$2^{105.7}$	$2^{142.3}$	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^{128}

Analysis of Complexity and Success Ratio 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 cleat that the complexity of *guess and solve* step is $2^l(\sum_{u_1=r_1}^{l-r_2} \sum_{u_2=r_2}^{l-u_1} mp_{u_1,u_2} + T_s)$, where T_s denotes the complexity of solving a

Algorithm 6 Attack on FiLIP Instantiated on XOR-THR-THR

```
Input: Number of fixing bits $r_1, r_2$, larger range $l$.
 1: $e \leftarrow \sum_{u_1=r_1}^{l-r_2} \sum_{u_2=r_2}^{l-u_1} mp_{u_1,u_2} 2^{-u_1-u_2} \sum_{v_1=r_1}^{u_1} \binom{u_1}{v_1} \sum_{v_2=r_2}^{u_2} \binom{u_2}{v_2}$
 2: $p_1 \leftarrow \frac{1}{2^{b_1-r_1}} \sum_{u_1=t_1-r_1}^{b_1-r_1} \binom{b_1-r_1}{u_1}$
 3: $p_2 \leftarrow \frac{1}{2^{b_2-r_2}} \sum_{u_2=t_2-r_2}^{b_2-r_2} \binom{b_2-r_2}{u_2}$
 4: $p \leftarrow \frac{1+(2p_1-1)*(2p_2-1)}{2}$
 5: **Data Filtering Phase:**
 6: $\mathcal{B} \leftarrow \emptyset$, randomly chosen $l$ indices forming $\mathcal{L}$
 7: **for** $i = 1$ **to** $m$ **do**
 8:     **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**
 9:         $\mathcal{B}$.append $(y_i, \sigma_i)$
10:     **end if**
11: **end for**
12: **Guess and Solve Phase:**
13: **for** each guess $x[\mathcal{L}] \in \{0,1\}^l$ **do**
14:     $\mathcal{G} \leftarrow \emptyset$
15:     **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**
16:         $\mathcal{G}$.append $(y_i = \mathsf{XOR}(x[\sigma_{i,1}] + w_i[1:a]) + 1)$
17:     **end if**
18: **end for**
19: $X \leftarrow$ System Solving $(\mathcal{G}, n-r, ep)$
20: **for** $x \in X$ **do**
21:     **if** $P(x[\sigma_i] + w_i | x[\mathcal{L}]) = y_i$ for all $i \in [1,m]$ **then**
22:         **return** $x$
23:     **end if**
24: **end for**
25: **return** $\perp$
```

group of correlation 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 $2^{-u_1} \sum_{v_1=r_1}^{u_1} \binom{u_1}{v_1} 2^{-u_2} \sum_{v_2=r_2}^{u_2} \binom{u_2}{v_2}$. Hence, the expected number of correlation 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 on $\mathsf{XOR\text{-}THR}$ predicate is

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

Identical to the attack in Section 5.1, Algorithm 6 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 4. The results demonstrate the efficiency of our attacks for most instances. For example, FiLIP cipher instantiated on $\mathsf{XOR}_{100}\text{-}\mathsf{THR}_{11,22}\text{-}\mathsf{THR}_{11,22}$ can be broken with no more than 2^{59} calls to Gaussian elimination, while the key size is 397 and the security assumption is $\lambda = 80$.

6 Conclusion

In this paper, we propose a new attack on Goldreich's PRGs with $\mathsf{XOR\text{-}THR}$ predicate. We exploit the bit-fixing correlation property and give attacks on the challenges which are immune to linear test and algebraic attacks. We give and implement an attack on an instance used for silent OT (EUROCRYPT 2024), which we implement on two PCs. Then we extend the attacks to FiLIP cipher instantiated on THR related predicates. The results show that $\mathsf{XOR\text{-}THR}$ is far from a well designed predicate that is immune to bit-fixing correlation attack. Based on these findings, the traditional security assumptions for Goldreich's PRGs, namely, (a) $\Omega(s)$-resilience and (b) algebraic immunity, are insufficient to guarantee pseudorandomness. Then the following problem remains open: whether a predicate that is simultaneously immune to linear test, algebraic attack and bit-fixing correlation attack exists?

References

1. 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). https://doi.org/10.1007/978-3-662-46800-5_17
2. 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). https://doi.org/10.1145/2213977.2214050
3. Applebaum, B.: Cryptographic hardness of random local functions - survey. Computational Complexity **25**(3), 667–722 (2016). https://doi.org/10.1007/s00037-015-0121-8

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). https://doi.org/10.1007/s00145-015-9202-8
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). https://doi.org/10.1007/978-3-319-63688-7_8
6. 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). https://doi.org/10.1007/978-3-031-30617-4_7
7. 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). https://doi.org/10.1145/2897518.2897554
8. 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). https://doi.org/10.1007/3-540-44647-8_1
9. 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). https://doi.org/10.1007/3-540-44647-8_1
10. 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). https://doi.org/10.1145/3133956.3134107
11. 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). https://doi.org/10.1007/978-3-031-58751-1_4
12. 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). https://doi.org/10.1007/978-3-030-03329-3_4

13. 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)
14. 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). https://doi.org/10.1007/978-3-319-63715-0_17
15. 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). https://doi.org/10.1007/978-3-031-64381-1_2
16. Goldreich, O.: Candidate one-way functions based on expander graphs. Electronic Colloquium on Computational Complexity (ECCC) **7**(90) (2000). https://doi.org/10.1007/978-3-642-22670-0_10
17. 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). https://doi.org/10.1145/2976749.2978332
18. 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). https://doi.org/10.1145/1374376.1374438
19. 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). https://doi.org/10.1007/978-3-319-70500-2_5
20. 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). https://doi.org/10.1007/978-3-030-35423-7_4
21. 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). https://doi.org/10.1007/978-3-662-49890-3_13
22. 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)
23. 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). https://doi.org/10.1109/CCC.2014.9
24. 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). https://doi.org/10.1145/2591796.2591825
25. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. SIAM J. Comput. **50**(3), 857–908 (2021). https://doi.org/10.1145/2591796.2591825
26. Wiedemann, D.H.: Solving sparse linear equations over finite fields. IEEE Trans. Inf. Theory **32**(1), 54–62 (1986). https://doi.org/10.1109/TIT.1986.1057137
27. 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). https://doi.org/10.1109/TIT.2021.3128315

A Proof of Lemma 1

Proof. For fixed $x_{i_1} = \cdots = x_{i_r} = 1$, $f(\cdot) = 0$ if and only if there are at least $t - r$ ones among the remaining $b - r$ bits. Hence,

$$p = \Pr[f(\cdot) = 1 | x_{i_1} = \cdots = x_{i_t} = 1] = \frac{1}{2^{b-r}} \sum_{i=t-r}^{b-r} \binom{b-r}{i}.$$

As a result,

$$\epsilon = p - \frac{1}{2} = \frac{1}{2}(2p - 1).$$

As

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

and

$$\binom{b-r}{i} = \binom{b-r}{b-r-i},$$

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

Consequently,

$$\epsilon = \frac{1}{2}(2p - 1) = \frac{1}{2^{b-r+1}} \sum_{i=t-r}^{(b-1)/2} \binom{b-r}{i}.$$

B Proof of Corollary 1

Proof. According to Lemma 1, the bias by fixing r bits to ones

$$\epsilon = p - \frac{1}{2} = \frac{1}{2^{b-r+1}} \sum_{i=t-r}^{(b-1)/2} \binom{b-r}{i}.$$

For each $i \in [t - r, (b - 1)/2]$,

$$\binom{b-r}{i} = \frac{(b-r)!}{i!(b-r-i)!}.$$

Substitute the Stirling's approximation,

$$\begin{aligned} \binom{b-r}{i} &= \frac{\sqrt{2\pi(b-r)}\left(\frac{b-r}{e}\right)^{b-r}}{\sqrt{2\pi i}\left(\frac{i}{e}\right)^{i}\sqrt{2\pi(b-r-i)}\left(\frac{b-r-i}{e}\right)^{b-r-i}} \\ &= \frac{\sqrt{b-r}}{\sqrt{2\pi i(b-r-i)}} \frac{(b-r)^{b-r}}{i^i(b-r-i)^{b-r-i}}. \end{aligned}$$

When $r \ll b$, $i \approx \frac{b-r}{2}$, so

$$\begin{aligned}\binom{b-r}{i} &\approx \frac{2^{b-r+1}}{\sqrt{2\pi(b-r)}} \\ &\approx \frac{2^{b-r+1}}{\sqrt{2\pi b}}.\end{aligned}$$

As a result

$$\frac{\binom{b-r}{i}}{2^{b-r+1}} \approx \frac{1}{\sqrt{2\pi b}}.$$

Hence,

$$\begin{aligned}\epsilon &= \frac{1}{2^{b-r+1}} \sum_{i=t-r}^{(b-1)/2} \binom{b-r}{i} \\ &\approx \frac{b-1}{2} - (\frac{b+1}{2} - r - 1)\frac{1}{\sqrt{2\pi b}} \\ &= \frac{r}{\sqrt{2\pi b}}.\end{aligned}$$

Then the correlation $p = \frac{1}{2} + \epsilon = \frac{1}{2} + \frac{r}{\sqrt{2\pi b}}$.