Vulnerability and Remedy of Stripped Function Logic Locking

Hai Zhou, Yuanqi Shen and Amin Rezaei
Northwestern University
haizhou@northwestern.edu, yuanqishen2020@u.northwestern.edu,
me@aminrezaei.com

Abstract. Stripped Function Logic Locking (SFLL) as the most advanced logic locking technique is robust against both the SAT-based and the removal attacks under the assumption of thorough resynthesis of the stripped function. In this paper, we propose a bit-coloring attack based on our discovery of a critical vulnerability in SFLL. In fact, we show that if only one protected input pattern is discovered, then the scheme can be unlocked with a polynomial number of queries to an activated circuit. As a remedy to this vulnerability, we also propose a provably secure general function that deregularizes the relation between the protected input patterns and the secret key. The mathematical proofs as well as the experiments confirm both the polynomiality of the bit-coloring attack on standard SFLL and the exponentiality of similar attacks on SFLL with general function.

Keywords: Logic Locking; SAT-based Attack; Stripped Function Logic Locking; One Way Function

1 Introduction

Increasing the design costs of Integrated Circuits (ICs) and growing the number of untrusted third party factories make chip protection one of the vital priorities for the semiconductor industry. Leakage of the IC layout to an insecure environment may lead to piracy and overproduction. In order to prevent the unauthorized products from functioning, logic locking [8, 5, 7, 3, 1] is proposed to insert key-controlled gates to the original design. To lock a circuit with a random n-bit secret key [8], first a combination of n buffers (for key bit "0") and inverters (for key bit "1") are chosen and matched with the bits of the key, and then each selected buffer or inverter is replaced with a key bit controlled XOR gate. In this case, the valid behavior of the circuit only happens when a correct key is applied. Such correct key

will be inserted in a tamper-proof memory by IC designer in post-fabrication phase.

However, almost all of the traditional logic locking schemes can be unlocked by the SAT-based attack [12] which efficiently deciphers the secret key with the help of an activated IC. Here, the assumption is that the attacker has full access to the physical layout of the locked circuit and also black-box access to the unlocked circuit. This can be easily obtained by buying two activated ICs from the market. One can be used for obtaining the physical layout of the locked circuit using powerful Reverse Engineering (RE) tools [13], and the other can be utilized as an oracle that tells the correct input-output pairs.

On the other hand, Stripped Function Logic Locking (SFLL) as an advanced locking technique [17] benefits from stripping an arbitrary part of the design functionality to provide a controllable resilience against the SAT-based attack. Upon inserting the correct key, the circuit is restored to the original one. The stripped function circuit can be captured in terms of input cubes for which the hardware-implemented design and the original one produce different outputs. These input cubes can also be conceived as conditions to manifest the built-in error. Therefore, an attacker applying the removal attack will obtain a netlist with this error with respect to the original design. However, any regularity in a logic locking design forms a signature that can be misused by the attackers. Although SFLL is robust against both the SAT-based and the removal attacks if the stripped function circuit is thoroughly resynthesised, it has a clear regularity in it: *All the protected input patterns are of the same Hamming Distance (HD) from the secret key.*

We argue that if an attacker knows only one protected input pattern, then she can decipher the correct key of SFLL with a straight forward polynomial time exact attack. On the other hand, if finding such protected input pattern is extremely hard for her, it means the built-in error is very low. Thus, the attacker can prototype an IC based on the stripped function circuit and

sell it on the market, constituting an approximate attack. The customers of this preliminary version can be rewarded to find any bugs. If a customer luckily finds out only one mismatch and reports it to the seller, the release version is ready by performing an exact attack. The contributions of this paper are threefold:

- Utilizing SAT solver to identify the HD parameter of SFLL in a single query;
- Proposing a novel bit-coloring attack that can decrypt SFLL with a linear number of queries to an activated IC;
- Suggesting a secure general function to deregularize the relation between the protected input patterns and the secret key in SFLL.

2 Background

The famous SAT-based attack [12] can efficiently unlock almost all of the traditional logic encryption methods [8, 5, 7, 3, 1] using a few input-output observations taken from an activated IC. The attack uses two copies of the locked circuit with the same input but different key values under a given constraint to check whether it is still possible to generate different outputs. Such input patterns are called Differentiating Input Patterns (DIPs.) The idea of using DIP is to exclude at least one wrong key from consideration. However, each DIP can exclude a large number of wrong keys in most cases. Thus, the main challenge for logic locking is to make it hard for an attacker figuring out a correct key by analysis of the locked circuit even if she has access to an activated IC.

After proposing the SAT-based attack, some counter measures [16, 14, 4] have been introduced. As an example, SARLock [16] adopts a comparator circuit to generate a flip signal that is asserted for specific input and key combinations. The flip signal will be XORed with one of the primary outputs. To prevent the flip signal from being asserted for the correct key

value, an additional mask logic is also inserted. In this case, SARLock ensures that each wrong key can only be excluded by checking one input pattern. Similarly, the main goal of the other SAT-proof methods like Anti-SAT [14] and AND-Tree [4] is to increase the required number of DIPs exponentially with the key size. Although these incremental techniques have high attack complexity, they suffer from very low error rate. Thus, they are vulnerable to approximate attacks that can return an almost correct key in which only a small number of input patterns produce wrong outputs.

Unlike the original SAT-based attack which returns an exact key, approximate attacks [11, 9, 15] relax the exactness constraint on decryption and return an approximate key. By deploying an approximate key in the circuit, an attacker can still make profit by selling the chip since a tiny number of wrong outputs can not be discovered immediately [10]. As an example, AppSAT [9] first uses the original SAT-based attack to prune the key values with a certain number of DIPs. Then, the SAT solver is utilized to provide a key value satisfying all these DIPs. To evaluate the correctness of the key value, random testing is adopted to estimate the error rate of the key. If the estimated error rate is below a specified threshold, the key value is considered as an approximate key. Otherwise, the random sampling that resulted in a disagreement will be added to the SAT formula as a new constraint. The combination of the SAT-based attack and random testing is repeated until the estimated error rate is below the threshold. Likewise, the other approximate attacks including Double DIP [11] and Bypass attack [15] also utilize the low error rate vulnerability of the SAT-proof methods to arrange a successful attack.

Sensitization attack as another oracle-based attack, determines individual key bits by generating and applying patterns that sensitize them to the outputs. In [6], two key bits are considered pairwise-secure if and only if the sensitization of one key bit cannot be done without controlling the other key bit

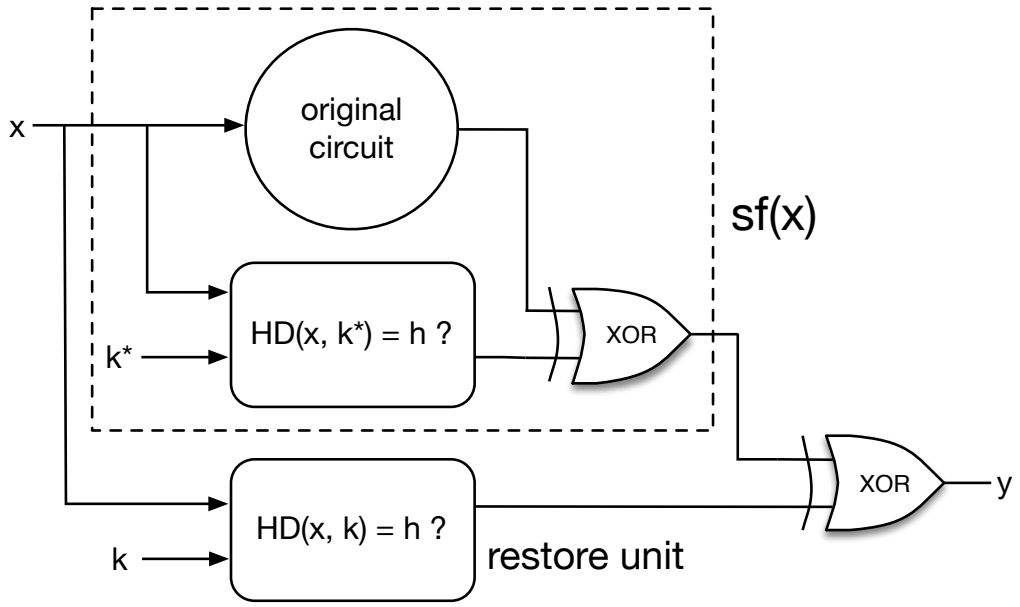

Figure 1: Stripped function logic locking

and vice versa. In addition, given a locked netlist, an attacker can identify and remove the lock blocks, organizing a removal attack.

Different from traditional SAT-proof methods, an advanced locking technique called SFLL [17] is proposed to strip part of the design functionality from its hardware implementation as shown in Figure 1. In this case, the design implemented in hardware is different from the original circuit, as the former will be missing the stripped functionality. Upon inserting the correct key, the stripped function circuit (i.e., $sf(x)$) is restored to the original one. SFLL is secure agains both the SAT-based and sensitization attacks. Also, in order to prevent removal attack, only $sf(x)$ needs to be obfuscated.

3 SFLL HD Parameter Detection

As a pre-step to attack SFLL, we introduce two approaches to find the hamming distance h. The first approach assumes one protected input pattern is known while the second one relaxes this assumption.

3.1 Utilizing Restore Unit

If a protected input pattern is known, the CNF formula of the restore unit shown in Figure 1 can be put into a SAT solver. To find h, we can constrain the output of the restore unit to be "1" and find assignments of x and k (i.e., $\hat{x}$ and $\hat{k}$) by a SAT query. Such $\hat{x}$ and $\hat{k}$ should satisfy the condition that $h = HD(\hat{x}, \hat{k})$. Therefore, h can be found by only one SAT query.

3.2 Utilizing Activated IC

Even without knowing a protected input pattern, h can be found by one query of the SAT-based attack [12], $C(x, k_1, y_1) \wedge C(x, k_2, y_2) \wedge (y_1 \neq y_2)$. Since $sf(x)$ in each copy generates the same output under the same x and k^*, in order to have an assignment satisfying $y_1 \neq y_2$, $\hat{x}$ must have a hamming distance h with either $\hat{k_1}$ or $\hat{k_2}$. We evaluate the correct output under $\hat{x}$ (i.e., $\hat{y}$) by an activated IC and compare $\hat{y}$ with $\hat{y_1}$ and $\hat{y_2}$. If $\hat{y_1} \neq \hat{y}$, then we can consider $h = HD(\hat{x}, \hat{k_1})$. Otherwise, $h = HD(\hat{x}, \hat{k_2})$. Therefore, we have the following lemma:

Lemma 1. *Given a circuit locked with SFLL, h can be found in one SAT query.*

4 SFLL Decryption

After finding the HD parameter, we propose two attacks on SFLL. The first attack that is an approximate one assumes finding a protected input pattern is extremely hard while the other attack that is an exact one assumes one protected input pattern is known.

4.1 Approximate Attack on SFLL

The selection of h determines the error rate and resilience to the SAT-based attack. Assuming primary inputs have n bits, when h approaches "0" or n, the error rate becomes exponentially low.

Algorithm 1: Bit-coloring attack

Input: The SFLL circuit $C(x, k, y)$, a protected input pattern $\hat{x}$, and original function $f(x)$

$\hat{x}', \hat{k}' = SAT(restore\_unit(x, k) = 1)$;
$h = HD(\hat{x}', \hat{k}')$;
$c(\hat{x}_0) = red$;
$\hat{x}' = \hat{x}$ with $\hat{x}_0$ flipped;
for $\hat{x}_i$ *in* $\hat{x}$, $1 \leq i \leq n-1$ **do**
    $\hat{x}'' = \hat{x}'$ with $\hat{x}'_i$ flipped;
    if $f(\hat{x}'') = sf(\hat{x}'')$ **then**
        $c(\hat{x}_i) = red$;
    else
        $c(\hat{x}_i) = green$;
$k^* = \hat{x}$ with $\hat{x}_i$ flipped, $0 \leq i \leq n-1$, if $|c(\hat{x}_i)| = h$.

When h approaches $n/2$, SFLL reaches the maximum error rate. In this case, the resilience to the SAT-based attack becomes minimum. Therefore, one suggested strategy [17] is to set h as $n/4$. However, such h still leads to low error rate. Therefore, an attacker can extract and build the $sf(x)$ part shown in Figure 1 as an approximate attack and sell such circuits to the market.

4.2 Exact Attack on SFLL

If a protected input pattern $\hat{x}$ s.t., $sf(\hat{x}) \neq f(\hat{x})$ is known, can an attacker quickly find the correct key k^* using the SAT-based attack [12]? The answer is no. Even by putting constraints $C(\hat{x}, k_1, f(\hat{x})) \wedge C(\hat{x}, k_2, f(\hat{x}))$ into the SAT solver, the size of remaining keys that cannot be pruned is still exponential.

Such observation leads to an interesting question: given one such $\hat{x}$, can we develop an attack to defeat SFLL in polynomial time? The answer is yes. The flowing theorem indicates the correct k^* can be found with only $n-1$ queries to $f(x)$:

Theorem 1. *Given $\hat{x}$ s.t., $sf(\hat{x}) \neq f(\hat{x})$, the correct key k^* of SFLL can be found in $n-1$ queries to an activated IC.*

Proof. We want to color each bit of $\hat{x}$ with green and red colors in a way that:

$$c(\hat{x}_i) = \begin{cases} red, & \text{if } \hat{x}_i \neq \hat{k}_i^* \\ green, & \text{otherwise} \end{cases}$$

Since k^* is unknown, we develop an attack method to find k^* by flipping bits of $\hat{x}$. We first flip the first bit of $\hat{x}$, then flip a bit $\hat{x}_i$ among the remaining $n-1$ bits to have $\hat{x}'$, and check whether $f(\hat{x}') \neq sf(\hat{x}')$. Please note that $HD(\hat{x}', \hat{x}) = 2$ and, if $f(\hat{x}') \neq sf(\hat{x}')$, $HD(\hat{x}', k^*) = h$. Therefore, the flipped bit $\hat{x}_i$ will be colored differently from the first bit of $\hat{x}$. We flip $\hat{x}_i$ back, and test the next bit. With these operations, we partitioned the bits of $\hat{x}$ into two groups. Because h is already known, we compare the number of bits in partitioned groups with h, and it is easy to show that at least one of the groups have h number of bits. Therefore, k^* can be found by flipping bits of $\hat{x}$ in that group. □

If the number of bits in the two groups are equal, we have two choices of k^*. Fortunately, the following lemma indicates these two choices are equivalent.

Lemma 2. *If the number of bits in two groups are the same after the coloring process illustrated in Theorem 1, the two choices of a correct key k^* are equivalent.*

Proof. Such a situation happens when $h = n/2$. Therefore, we can either find k^* or its complement of $\overline{k^*}$. If k^* and $\overline{k^*}$ share the same protected input patterns, then k^* and $\overline{k^*}$ are functionally equivalent.

Assuming $\exists\ \hat{x}$ s.t., $HD(\hat{x}, k^*) = h$ but $HD(\hat{x}, \overline{k^*}) \neq h$. Since $\overline{k^*}$ is the complement of k^*, $HD(\hat{x}, \overline{k^*}) = n - h$. Since $h = n/2$, $HD(\hat{x}, k^*) = HD(\hat{x}, \overline{k^*}) = n/2$, which is a conflict with the assumption. □

Here we provide an example. Assuming $k^* = 10110$, $h = 2$, and an attacker has a protected input pattern $\hat{x} = 00010$. We flip the first bit of $\hat{x}$ to have 10010. We then flip individual bits from $\hat{x}_1$ to $\hat{x}_4$. The result of coloring is shown in Table 1. Since

Table 1: Example of the coloring process

flip	$\hat{x}'$	$sf(\hat{x}') \neq f(\hat{x}')$?	$c(\hat{x}_i)$
$\hat{x}_1$	11010	yes	$\neq c(\hat{x_0})$
$\hat{x}_2$	10110	no	$= c(\hat{x_0})$
$\hat{x}_3$	10000	yes	$\neq c(\hat{x_0})$
$\hat{x}_4$	10011	yes	$\neq c(\hat{x_0})$

$h = 2$, we know that the group with two bits ($\hat{x_0}$ and $\hat{x_2}$) has the same color. Therefore, k^* can be simply found by flipping $\hat{x}_0$ and $\hat{x}_2$.

Algorithm 1 shows complete procedure of the bit-coloring attack. Given a protected input pattern, h can be found with only one SAT query, and k^* can be deciphered with $n-1$ queries to an activated IC.

5 SFLL Enhancement

The current SFLL design can be enhanced to avoid the bit-coloring attack. Instead of directly comparing primary inputs x and key inputs k, one straightforward solution is to include a One Way Function (OWF) that can be utilized to generate $O(k)$ as shown in Figure 2. In practice, OWF can be substituted by a cryptographic hash function such as MD5 or SHA. OWF improves the original SFLL design since the key values are hidden, and the security level can be proved in Theorem 2.

Theorem 2. *Finding the correct key k^* in Figure 2 is equivalent to invert the one way function $O(k)$.*

Proof. An attacker may still find outputs of the $O(k)$ in $sf(x)$, $\hat{v}$, by conducting the bit-coloring attack on $sf(x)$. If an attacker finds k^*, an attacker successfully finds an input k^* s.t. $O(k^*) = \hat{v}$, which inverts the one way function $O(k)$. On the other hand, if an attacker can invert the one way function $O(k)$, she can easily find k^* if $\hat{v}$ is known. Therefore, finding the correct key

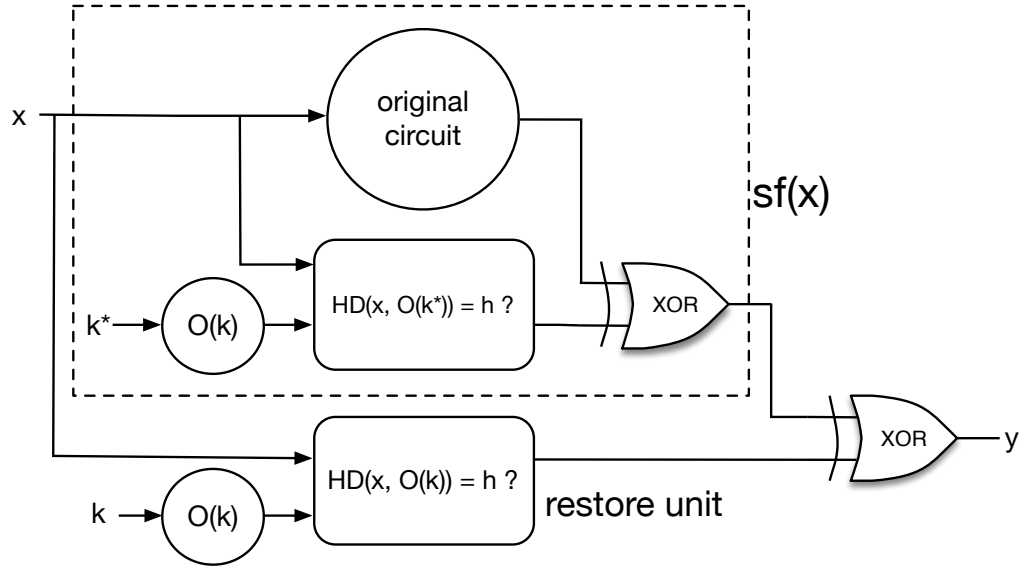

Figure 2: The design of SFLL with one way functions O(k)

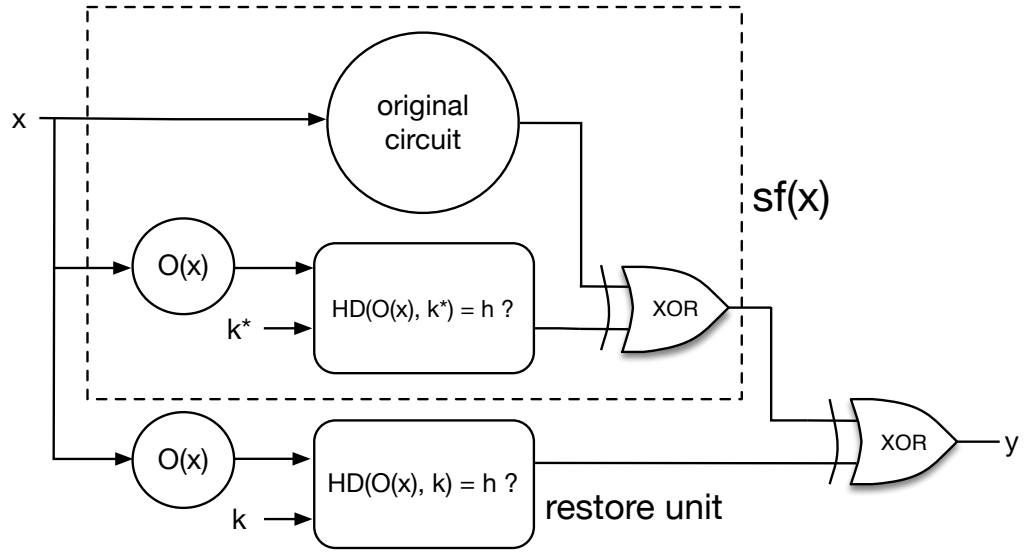

Figure 3: The enhanced design of SFLL with one way functions O(x)

k^* in Figure 2 is equivalent to invert the one way function $O(k)$, which is with negligible probability. □

Theorem 2 indicates that the correct key value k^* in Figure 2 is extremely difficult to find by an attacker. However, such a design is still not fully secure. Specifically, an attacker may decrypt such encryption without knowing k^*. An attacker may still locate outputs of the $O(k)$ connecting to the restore unit,

remove the $O(k)$ and consider its outputs as a new key value. Therefore, she can bypass $O(k)$, conduct the bit-coloring attack, and find new key values again. Thus, we propose a better SFLL design as shown in Figure 3. In this design, OWFs are inserted in the way of the primary inputs.

Such design is resilient to the bit-coloring, the SAT-based, sensitization, and removal attacks as shown in Theorem 3 to Theorem 6. We assume the length of primary inputs, the output of the one way function, and key inputs are n, m and k, respectively, where $k \leq n \leq m$. We assume the one way function is collision free since its collision-resistance is usually exponentially large.

Theorem 3. *The enhanced SFLL design shown in Figure 3 is with negligible probability to be decrypted by the bit-coloring attack within polynomial time.*

Proof. The OWFs are with the property that it is hard to find two inputs $\hat{x}$ and $\hat{x}'$ s.t. $O(\hat{x}) = O(\hat{x}')$, and $\hat{x} \neq \hat{x}'$. Therefore, it is with negligible probability to flip x_0 and find another x_i, $1 \leq i \leq n-1$, to cancel out the variation of hamming distance within polynomial time. □

Theorem 4. *The enhanced SFLL design shown in Figure 3 is* $(k - log_2\binom{k}{h})$*-secure against the SAT-based attack.*

Proof. The original SFLL design shown in Figure 1 is $(k - log_2\binom{k}{h})$-secure against the SAT-based attack [17]. Similarly, to decrypt the enhanced SFLL design, an attacker can randomly guess to find a protected input pattern with a probability $2^{m-k}\binom{k}{h}/2^m = \binom{k}{h}/2^k$. Therefore, with q queries, the success probability is $q\binom{k}{h}/2^k = q/2^{k-log2\binom{k}{h}}$. The enhanced SFLL design is also $(k - log_2\binom{k}{h})$-secure against the SAT-based attack. □

Theorem 5. *The enhanced SFLL design shown in Figure 3 is* k*-secure against sensitization attack.*

Proof. Since OWFs do not directly affect key values, similar to the proof in [17], an attacker need to control all other key bits to

sensitize a key bit to the output of the restore unit. Therefore, all k bits are pairwise-secure, and the enhanced SFLL design is k-secure against sensitization attack. □

Theorem 6. *The enhanced SFLL design shown in Figure 3 is* $2^{n-k}\binom{k}{h}$*-resilient to removal attack.*

Proof. An attacker can still extract $sf(x)$. However, there are $2^{m-k}\binom{k}{h}$ inputs to the hamming distance block in $sf(x)$ such that $f(x) \neq sf(x)$. Since $n \leq m$, $sf(x)$ is still with $2^{n-k}\binom{k}{h}$ protected input patterns. Therefore, The enhanced SFLL design is $2^{n-k}\binom{k}{h}$-resilient to removal attack. □

6 Experimental Results

In this section, we evaluate the correctness and the efficiency of the bit-coloring attack. Our experiments are conducted on a machine with 2.4 GHz Intel Core i5, running Linux with memory 8 GB. We randomly select the correct key k^* with different lengths, and we choose different values of h for each k^*. Table 2 indicates the result. k^* can be found with all the test cases, and the execution time is around 1 second for all the combination of k^* and h. The experimental results show that if a protected input pattern is given, the correct key k^* can be found in very short time. However, this cannot be done in linear time by the SAT-based attack.

Meanwhile, we adopt the Goldreich′s OWF [2] as the $O(x)$ function. The experiment shows that the enhanced SFLL encryption shown in Figure 3 cannot be decrypted in polynomial time by both the bit-coloring and the SAT-based attacks. The bit-coloring attack can be finished quickly, however, a correct key k^* is not solved.

Table 2: Bit-coloring attack efficiency

Bits of k^*	h	Correctness	Time(s)
64	16	yes	1.1
64	32	yes	0.6
64	48	yes	1.3
64	64	yes	0.6
128	32	yes	1.2
128	64	yes	0.6
128	96	yes	0.5
128	128	yes	0.5
256	64	yes	1.1
256	128	yes	0.5
256	192	yes	0.5
256	256	yes	0.5
512	128	yes	1.1
512	256	yes	0.6
512	384	yes	0.6
512	512	yes	0.5
1024	256	yes	0.9
1024	512	yes	0.6
1024	768	yes	1.2
1024	1024	yes	0.6

7 Conclusion

Recently, SFLL has been proposed to successfully defeat both the SAT-based and removal attacks. In this paper, we carefully analyzed the security of SFLL. First, we suggested two smart methods to find the hamming distance h by one SAT query. Then, we utilized the regularity between the protected input patterns and the secret key to propose an efficient bit-coloring attack. The attack can decipher the secret key with $n-1$ queries. The experimental results show that given a protected input pattern, the correct key can be found in a very short time by the bit-coloring attack. We further suggested an enhanced SFLL design with OWFs to defeat the bit-coloring attack.

References

[1] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing ic piracy using reconfigurable logic barriers. In *IEEE Design Test of Computers*, volume 27, Issue 1, pages 66–75, 2010.

[2] O. Goldreich. Studies in complexity and cryptography. pages 76–87. Springer-Verlag, 2011.

[3] K. Juretus and I. Savidis. Reduced overhead gate level logic encryption. In *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 15–20, 2016.

[4] M. Li, K. Shamsi, T. Meade, Z. Zhao, Bei B. Yu, Y. Jin, and D. Z. Pan. Provably secure camouflaging strategy for ic protection. In *International Conference on Computer-Aided Design (ICCAD)*, 2016.

[5] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Logic encryption: A fault analysis perspective. In *Design, Automation and Test in Europe (DATE)*, pages 953–958, 2012.

[6] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Security analysis of logic obfuscation. In *Design Automation Conference (DAC)*, pages 83–89, 2012.

[7] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. Fault analysis-based logic encryption. In *IEEE Transactions on Computers*, volume 64, issue 2, pages 410–424, 2015.

[8] J. A. Roy, F. Koushanfar, and I. L. Markov. Epic: Ending piracy of integrated circuits. In *Design, Automation and Test in Europe (DATE)*, pages 1069–1074, 2008.

[9] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin. Appsat: Approximately deobfuscating integrated circuits. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 95–100, 2017.

[10] Y. Shen, A. Rezaei, and H. Zhou. A comparative investigation of approximate attacks on logic encryptions. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 271–276, 2018.

[11] Y. Shen and H. Zhou. Double dip: Re-evaluating security of logic encryption algorithms. In *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 179–184, 2017.

[12] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 137–143, 2015.

[13] R. Torrance and D. James. The state-of-the-art in semiconductor reverse engineering. In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 333–338, 2011.

[14] Y. Xie and A. Srivastava. Mitigating sat attack on logic locking. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 127–146, 2016.

[15] X. Xu, B. Shakya, M. Tehranipoor, and D. Forte. Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 189–210. Springer, 2017.

[16] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu. Sarlock: Sat attack resistant logic locking. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 236–241, 2016.

[17] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu. Provably-secure logic locking: From theory to practice. In *Conference on Computer and Communications Security (CCS)*, pages 1601–1618, 2017.