A Humble Theory and Application for Logic Encryption

Hai Zhou
Northwestern University

1 INTRODUCTION

Logic encryption (or logic locking) is a technique to manipulate a given combinational circuit with added key input to make sure that the encryption circuit will only function as the original one under a specific key value and it is difficult to figure out that value. It is an important problem for IP protection, IC production control, Trojan prevention, and many other applications in hardware security. Circuit camouflaging, where ambiguity is intentionally introduced in the layout to fool the reverse-engineer, can also be modeled as logic encryption with keys to encode the different possibilities [15, 18].

Even though there exist many different approaches for logic encryption [3, 10, 17, 19, 20], all of them are based on ad hoc approaches to insert extra gates with keys to the original circuit. Therefore, it should not be too great a surprise (even though it was actually a surprise to many people) that a SAT-based attack developed by Subramanyan et al. [24] can efficiently decrypt almost all of the encrypted circuits by them.

Immediately after Subramanyan et al. [24], some remedies were proposed to strengthen the existing logic encryption. Yasin et al. [28] proposed SARLock, which was inspired by the difficult case of the AND-tree discovered in [24], and ensures that each wrong key can only be excluded by one input. Xie and Srivastava [27] developed the Anti-SAT encryption, where one key has at most one wrong input, but one input may exclude many wrong keys. However, all these remedies have extremely low error rate; both SARLock and Anti-SAT have 2^{-n} error rate (i.e. one input is wrong on each wrong key). Therefore, to protect against random guess attack, they have to be combined with traditional encryption methods.

The remedies such as SARLock and Anti-SAT combined with traditional encryptions make any SAT-based exact attack (i.e. getting the correct key) exponentially expensive. However, it may be vulnerable to approximate attacks that can return a key with very low error rate. Double DIP [23] and AppSAT [21] are the first approaches for approximate attacks to logic encryption.

Conference'17, July 2017, Washington, DC, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this paper, we would like to change the status quo of logic encryption by developing a theory for it. We start with a basic understanding that there are two entangled goals in logic encryption: locking and obfuscation. *Locking is a logical request to make sure that the correct behavior only happens when a correct key is applied and the correct key cannot be easily figured out by studying the* logic *of the encryption circuit. Obfuscation is a structural request to make sure that the correct circuit cannot be revealed by any* structural *analysis of the encryption circuit.*

We suspect that the entangled goals of locking and obfuscation might be one cause of current ad hoc approaches in logic encryption. Therefore, we want to separate the two concerns starting from the beginning. By investigating the logic of all possible encryptions for a given function, we develop a theory for logic encryption that captures the whole design space and the relation between a design and its attack complexity. With error rate as a design goal, we also establish the relation between error rate and attack complexity. Both minimal and average attack complexities are considered in our work, and what we discover is that there is a contention between attack complexity and error numbers and their product cannot be larger than $n2^n$, even for average attack complexity.

Our theory has been initially investigated without any concern on the size of the encryption circuits. Fortunately, when the size constraint is considered, we find that the above bound can still be realized by *linear-sized* encryption circuit. The benefit comes from the modulating of the keys by the normal inputs with XOR gates. Based on this, we propose a general logic encryption scheme.

In order to further thwart SAT-based logic analysis attacks, we also propose to insert Goldreich's one-way function [12] in the scheme to increase the running time for SAT engines.

We should caution the readers not to worry about the easiness of structural attack to our proposed logic encryption scheme, since it represents only the logic functionality of any equivalent circuits. Circuit obfuscation will always be applied to these logic to hide any structural vulnerability.

Circuit (or program) obfuscation has been a practice in software engineering for a long time. However, its theoretical study only started relatively recently [2]. Even with a recent breakthrough on indistinguishability obfuscation [11], it is still too expensive to be deployed in our logic encryption. We

will show that resynthesis is complete for *the best-possible obfuscation* [13], and propose to do resynthesis on our logic encryption to protect it from structural analysis attacks.

It is very difficult to measure the performance of an approximate attack, since the error rate cannot be measured on the number of key bits that the same as the correct one, and exactly measuring of error rate requests to simulate all input combinations. As an application of our theory, we develop a suite of scientific benchmarks for approximate attacks. On these benchmarks, different key has different error rate, which can be simply calculated based on the key.

We also test our logic encryption designs and their obfuscations with Goldreich's one-way function by the SAT-based attack [24]. The results confirms our theory and the robustness of our approach.

2 BACKGROUND AND PROBLEM FORMULATION

2.1 Conventional Logic Encryption and SAT-Based Attack

There exist many different ways [3, 10, 17, 19, 20] for logic encryption. However, they are all based on the general idea of iteratively find a signal at random in the original circuit and insert a lock gate (mostly an XOR) with a key. Examples are given in Figure 1. Of course, they also try to prevent circuit analysis and simple testing-based attacks by carefully selecting the lock gate locations and doing resynthesis afterwards.

An attacker is generally assumed to have access to the encryption circuit either by reverse-engineering or through other ways. He is also assumed to have a blackbox access to the original circuit, for example, through product purchase on the market. Since almost all product ICs are sequential circuits, the combinational circuit assumption we use here assumes an access to the scan-chain.

With this attack model in mind, Subramanyan et al. [24] proposed a SAT-based attack that can effectively defeat almost all of the traditional logic encryption methods. We first give its pseudo-code here in Algorithm 1. The main step in the SAT-based attack is to use two copies of the encryption circuit with the same input but different keys under a given constraint to check whether it is still possible to generate different outputs. Such input patterns are called Differentiating Input Patterns (DIPs). Each DIP is then used to query the original circuit blackbox to get the correct output. The DIP with output is then used to further constrain the keys under consideration.

Algorithm 1 SAT Attack Algorithm

Require: An encryption circuit $g(x, k)$ and original boolean function $f(x)$.
Ensure: Correct key k^* such that $g(x, k^*) \equiv f(x)$.
1: **while** $\hat{x} = SAT(g(x,k) \neq g(x,k1))$ **do**
2: $\quad \hat{y} = f(\hat{x})$;
3: $\quad g(x,k) = g(x,k) \wedge (g(\hat{x},k) = \hat{y})$;
4: $\quad g(x,k1) = g(x,k1) \wedge (g(\hat{x},k1) = \hat{y})$;
5: **end while**
6: $k^* = SAT(g(x,k))$;

The idea of using DIP is to exclude at least one wrong key from consideration. However, the surprise is that many of the DIPs each may exclude a large number of wrong keys. That is the main reason for the effectiveness of the attack. Of course, if a DIP can only exclude a very small number of wrong keys, then the attack will take very long time to find the correct key. Yasin et al. [28] and Xie and Srivastava [27] explored this property to develop strengthening approaches. But since they both have an error rate of 2^{-n}, they can be effectively defeated by approximate attacks such as Double DIP [23] and AppSAT [21].

2.2 Problem Definition

The logic encryption problem can be formulated as follows. Given a Boolean function $f : B^n \rightarrow B$ represented by a multi-level netlist of Boolean gates, find a Boolean function $g : B^{n+m} \rightarrow B$ also as a multi-level netlist, such that

(1) There is a random Boolean m-vector k^* such that $g(x, k^*) \equiv f(x)$;
(2) The netlist size of g is only a polynomial of that of f;
(3) With the netlist of g and a black-box access to f, it is difficult to "discover" f.

The first two requirements are straightforward: the first one means there is at least one correct key to decrypt the logic; the second one only allows a polynomial blow-up of the circuit size. For performance critical applications, we may also request that the netlist depth of g be bounded. However, the last requirement is tricky and requests a thorough discussion.

There are two different aspects that need to be discussed in the third requirement: what it means to "discover" f, and how to measure the difficulty. The conventional definition for "discovering" f is to find k^* or any k such that $g(x, k) \equiv h(x)$. The interesting result of Subramanyan and Malik [24] was to find that almost all existing logic encryption algorithms before them can be effectively decrypted by a SAT-based attack. The effectiveness is measured by the number of inputs that are queried on the black-box of f, which is bounded by a polynomial of n. In addition, it also depends on the efficiency of solving a SAT problem on all these inputs for a key.

Immediately after the SAT-based attack by Subramanyan and Malik [24], there come a few remedies for it. SARLock [28] utilizes the AND-tree structure discovered in Subramanyan

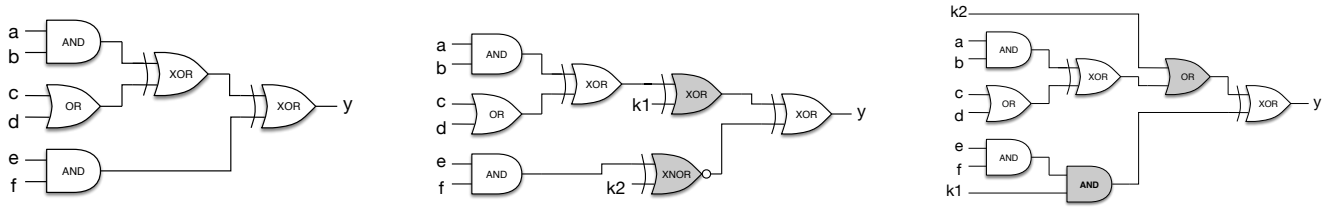

Figure 1: Traditional logic encryption example: gray gates are lock gates.

and Malik [24] to thwart the SAT-based attack by ensuring that the number of inputs to be queried is at least exponential. Anti-SAT [27] is another systematic way to design logic encryptions that will also request the SAT-based attack to query exponential number of inputs. One drawback for both SARLock and Anti-SAT approaches is that when applying an arbitrary key k, the error rate is extremely low. Here the error rate is defined as the percentage of inputs x that will generate wrong output, meaning, $g(x, k) \neq f(x)$. *For both SARLock and Anti-SAT, there is exact one input with wrong output for every $k \neq k^*$.*

The extreme low error rates in SARLock and Anti-SAT entice a new type of attacks: approximation attacks. A random guess k on the key will give a circuit $g(x, k)$ that is "almost correct" where a wrong result will happen on at most one input. The justification for such approximation attacks is that the attacker is very sure that the unauthorized circuit can be sold on the market with an extremely low probability of being caught. Furthermore, any catch of a discrepancy on a given input will help the attacker to correct the key.

Another argument for approximate attacks–assuming we are in the shoes of an attacker–is that, under a wrong key with extremely low error rate, the attacker now not only pirates on your intellectual property but also places in it a perfect Trojan horse. Remember that an extremely low error rate means that it will be extremely difficult to detect the Trojan. In this sense, we may argue that approximate attacks are even more malicious than the exact attacks, thus should be seriously prevented in logic encryption.

Obviously, Yasin et al. [28] and Xie and Srivastava [27], the designers of SARLock and Anti-SAT respectively, were already aware of the approximate attacks. Therefore, they both proposed in their work to combined the new encryption approaches with existing approaches, just in order to thwart the approximation attacks [21, 23].

With this perspective, we believe that it is necessary to include approximate attack as successfully "discovering" f in our problem formulation. When we argue that a logic encryption is difficult to decrypt, it is not sufficient to consider only existing attack methods such as SAT-based attack and 2DIP attack, since who knows what new attack method will be invented tomorrow. Following the tradition-honored practice in cryptography, we would better consider probabilistic learning as the most general form for "discovering" f. Therefore, we should consider decryption as finding a key k such that $g(x, k)$ is Probabilistically Approximately Correct (PAC) [25] for $f(x)$.

3 NOVEL LOGIC ENCRYPTION DESIGNS

Besides the attack complexity, another design criteria of encryption circuit is the error rate. For any wrong key, the error rate is the ratio of inputs generating wrong outputs. In other words, the error rate is the error number divided by the total number of possible inputs (2^n). All existing remedies [27, 28] against the SAT-based attack have extremely low error rate: 2^{-n}. Therefore, an interesting problem to investigate is whether there could be logic encryptions that have both exponential SAT attack complexity and substantial error rates.

In our system, the error number for k is given by

$$error(k) \triangleq \sum_{i=0}^{2^n-1} g(i, k) \neq f(i).$$

We can also symmetrically define the error number for each input i as

$$error(i) \triangleq \sum_{k=0}^{2^m-1} g(i, k) \neq f(i),$$

which can be measured as the number of minterms of k not included in function $g(i, k) = f(i)$. Even though requesting $error(k)$ to be exponential for every wrong k is different from requesting $error(i)$ to be exponential for every input i. It can be seen that they are closely related since the sum is the same, that is,

$$\sum_{i=0}^{2^n-1} error(i) = \sum_{k=0}^{2^m-1} error(k).$$

More importantly, we can also view the logic encryption design as to determine the following error matrix.

$(g(x,k) \neq f(x)) =$

$$\begin{array}{c} \left[\begin{array}{ccccccc} 1 & \dots & 0 & \dots & 0 & \dots & 1 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & g(i,j) \neq f(i) & \dots & 0 & \dots & 1 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \dots & 1 & \dots & 0 & \dots & 0 \end{array}\right] \begin{array}{c} \text{x=0} \\ \vdots \\ \text{i} \\ \vdots \\ 2^n-1 \end{array} \\ \begin{array}{ccccccc} \text{k=0} & \dots & \text{j} & \dots & \text{k}^* & \dots & 2^m-1 \end{array} \end{array}$$

Each row represents a given x value from 0 to $2^n - 1$, and each column represents a given k value also from 0 to $2^n - 1$. The element at (i, j) represents the value of $g(i, j) \neq f(i)$, that is, an error at $g(i, j)$. The error number for k is the number of ones in column k. The minimal error number is the minimal number of ones in any column with one. On this error matrix, the decryption problem can be formulated as a covering problem: a subset of rows (inputs x) are sufficient if and only if they cover all the columns with ones.

Such a unate covering as optimization problem is well studied in logic synthesis [9]. It is also known as the *set cover, hypergraph covering, hitting set,* or *hypergraph traversal* problem. However, our goal here is not optimization. We want to design a matrix such that the minimal number of ones for any column with one is large, while the number of rows needed to cover them is also large. If the minimal error number is M, any error number larger than M on any key will only decrease the attack complexity. Therefore, we will first focus on a setup where every wrong key has exactly M errors. We call it the *uniform error number* logic encryption. The corresponding hypergraphs are called *M-graphs* where each hyper-edge has M vertices.

The minimal cover of the M-graph corresponding to the error matrix provides a lower bound on the attack complexity, even though achieving such a lower bound is almost impossible. Since we are looking for a design to maximize the attack complexity, we are interested in an M-graph that maximizes its minimal cover. However, Alon [1] has shown that the cardinality of the minimal cover for any M-graph is upper bounded by $(1 + o(1))\frac{\ln M}{M}(2^n + 2^m)$, and Chvátal and McDiarmid [7] have given another upper bound of $(2^n + \lfloor M/2 \rfloor 2^m)/\lfloor 3M/2 \rfloor$ [1]. These results indicates that there is a natural contention between the minimal attack complexity and the minimal error number in the logic locking.

Since it is extremely unrealistic to expect that the SAT-based attack will discover the minimal cover in the matrix, let us investigate whether we can escape from this contention if the average attack complexity is considered instead of the minimal attack complexity. Unfortunately, we cannot do much, because of the following lemma.

Lemma 3.1. *In any given encryption $g(x, k)$ for any function $f(x)$, if the minimal error number for any wrong key is M, then $m2^n/M$ random DIP queries will decrypt it with a probability at least $1 - (2/e)^m$.*

Proof. We will consider a sequence of N independent random selection of rows and to calculate the probability that they are still not a cover. Please note that a DIP selection in SAT-based attack is dependent on existing selections and also no repeated selection is allowed. Therefore, such a probability for independent random selection is an upper bound of the probability for dependent DIP selections.

Now consider each column with one in it. It must have at least M ones. Therefore, an independent random selection of a row will not cover it with a probability at most $1 - M/2^n$. A sequence of N selections will not cover it with a probability at most $(1 - M/2^n)^N$. There are at most $2^m - 1$ such columns, thus, such selections will not form a cover with a probability at most $2^m(1 - M/2^n)^N$. It can be shown that $(1 - M/2^n)^{2^n/M}$ is monotonically increasing and converges to e^{-1}. Therefore, if $N = m2^n/M$, then

$$2^m(1 - M/2^n)^N \leq 2^m e^{-m} = (2/e)^m.] \quad \square$$

Fortunately, we can show that there does exist an encryption with both high attack complexity and high error numbers. The following lemma is just a direct application of the results in hypergraph covering [1, 7].

Lemma 3.2. *For any given $f(x)$, there exists a logic encryption $g(x, k)$ with M as the minimal error number whose minimal attack complexity is close to the bounds given by Alon [1] and Chvátal and McDiarmid [7].*

But such a construction based on hypergraph is totally impractical, since the provided $g(x, k)$ most possibly must have exponential circuit size. The more important problem is to find such a circuit with a small size. Based on our theory, each function $g(i, k) = f(i)$ has to distinguish (thus to exclude) an exponential number of minterms. The request of small size on g forbids one different block for every $g(i, k) = f(i)$, otherwise there will be exponential number of blocks. Without loss of generality, let $g(0, k) = (m_{k*} + h(k)) \equiv f(0)$ be the block shared by every $g(i, k)$ for $i \in 0..2^n - 1$. A good (or perhaps the best) way to get distinguished minterms from each $g(i, k)$ is to modulate k in $h(k)$ bit-wisely by i, that is, to make

$$g(i, k) = (m_{k^*} \vee h(k \oplus i)) \equiv f(i).$$

In this case, we can have a general design as shown in Figure 2, which is also given by the following formula,

$$g(x, k) = (m_{k^*}(k) \vee h(k \oplus x)) \equiv f(x).$$

[1] Thanks to Shamsi et al [22], an error in the first version of the paper has been corrected here.

Its correctness is stated in the following theorem.

THEOREM 3.3. *If function $h : B^n \to B$ has an on-set of size M, then the logic encryption given in Figure 2 will have an error number of M for every wrong key, and a minimal attack complexity at least $2^n/M$.*

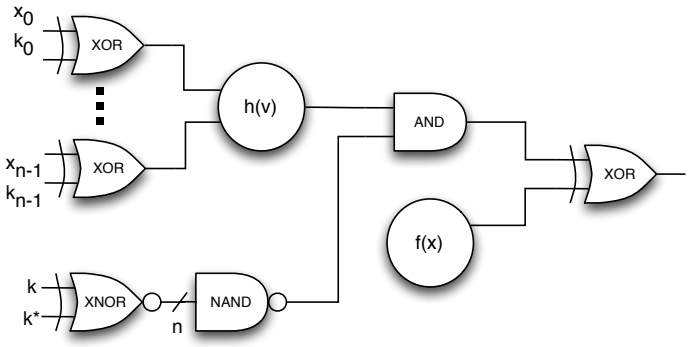

Figure 2: A general logic encryption scheme where h(v) has an on-set of size M.

A simple design following the general scheme is to have $h(v) = \bigwedge_{i=0}^{n/2-1} v_{2i} \oplus v_{2i+1}$, as shown in Figure 3. It can be seen that the on-set of the h function in this design is $2^{n/2}$. Therefore, it has $2^{n/2}$ as the error number for every wrong key and at least $2^{n/2}$ for the minimal attack complexity.

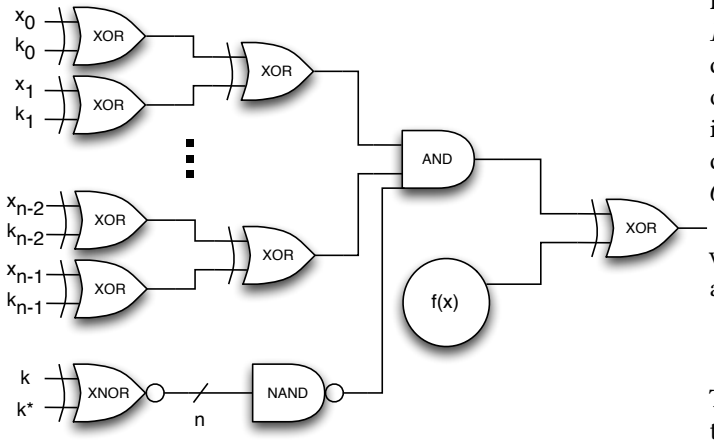

Figure 3: A simple logic encryption design with both exponential attack complexity and error number.

4 INSERTION OF ONE-WAY FUNCTION

Increasing the number of necessary iterations in the SAT-based attack is just one way to increase the attack complexity. Another way is to increase the complexity of SAT instances in the DIP finding. For this purpose, we need to look for hard instances for SAT problem and integrate them into the logic encryption circuit. Cryptography is one of the promising areas to look for hard SAT instances.

Similar ideas have been proposed and discussed by Yasin et al. [29] and cited by Xie and Srivastava [27]. However, they only proposed to use AES as the hard instance. It is well known that AES is a complicated algorithm, with at least 10 cycles of repetition even for the smallest 128-bit key configuration. To be used in the encryption circuit, which is a combinational circuit, the AES has to be unrolled to make it combinational, which will definitely increase its size. The result in [29] only showed the execution time of the attack, but not the circuit size of inserted AES. But it can be estimated that AES will introduce substantial overhead on the circuit size.

We advocate here to use Goldreich's candidate one-way functions based on expander graphs [12] as the hard instances inserted in logic encryptions. The benefits include:

- The Goldreich one-way functions are naturally combinational thus no unrolling is needed;
- They are simple to implement and have only linear sizes in terms of the input size;
- Their one-wayness has been thoroughly investigated and experimentally confirmed with SAT engines [8].

Goldreich's one-way functions are easy to construct. There are two parameters to select: a connection degree d and a predicate P on d inputs. For any n-bit inputs, the one way function will compute each bit of its output by applying P on a random selection of d input bits. There are some criteria to follow: P should not be linear sum or degenerate on the inputs; if the connection between inputs and outputs is treated as a bipartite graph, it has to be an expander. The connection degree d can be very small, in $O(\log n)$ or even $O(1)$.

Cook et al. [8] had a thorough study on Goldreich's one-way functions. Based on previous study, they even suggested a simple predicate

$$P(x_0, \ldots, x_{d-1}) = x_0 \oplus x_1 \ldots \oplus (x_{d-2} \wedge x_{d-1}).$$

They have conducted experiments with SAT engines on functions thus constructed. Even with $d = 5$, they observed an exponential increase of running time as a function of the input length n. Their experiments also indicate that the MiniSat engine will take more than 10 seconds if the input length is 140. That provides a strong evidence for us to suggest such functions to be inserted in logic encryption.

The next question then is where and how to insert Goldreich's functions. There are two possible locations for inserting the function in our general logic encryption scheme shown in Figure 2. One is before the h function, and the other is before the key input to the circuit.

The advantage of the first location is that the function is mixed up well with the rest of the encryption circuit. Its drawback is that, combined with this random function, it is hard to calculate the error rate and attack complexity of the encryption. However, if we assume that these random one-way functions have low collision rate (a collision means two different input mapped to the same output), the disturbance to the error rate and attack complexity will be tiny.

On the other hand, even though the second location (before the key input) will not affect the h function or its on-set, it only give us the benefit of accurately calculating the attack complexity. The error rate with respect to the output of the one-way function is known, but it is still unknown with respect to the input of the one-way function. The drawback of the second location is its not mixing up with other part of encryption circuit. Even though we will definitely do obfuscation by resynthesis of the whole encryption circuit, as discussed in the next section, thus can mix the one-way function with other part of the encryption, its benefit may not justify its adaptation.

Therefore, we suggest to use the general logic encryption with one-way function as shown in Figure 4. Here, the G function is Goldreich's one-way function. The simplest design could use $d = 5$ with

$$P(x_0, \ldots, x_4) = x_0 \oplus x_1 \oplus x_2 \oplus (x_3 \wedge x_4).$$

Please note that we do not need to have a big one-way function to make the SAT engine infeasible to solve one iteration in the attack. Remember that with one more iteration in the SAT-based attack, one instance of the encryption circuit will be added with one DIP into the CNF. Therefore, even though one instance of the one-way function may not be too difficult to solve, its effect will accumulate with each more iteration, and make it getting slower and slower. Remember that our design requests at least exponential number of iterations to decrypt.

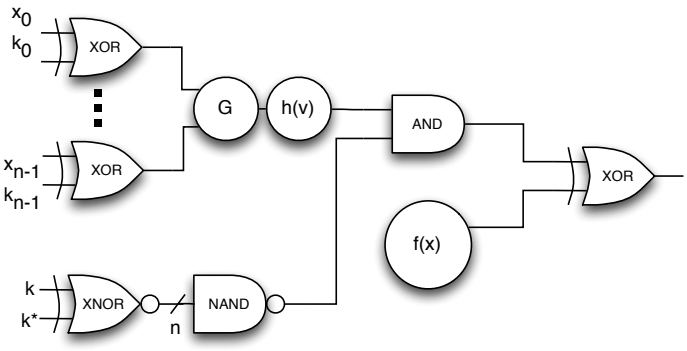

Figure 4: A general logic encryption with one-way function, where G is Goldreich's one-way function, and h(v) has an on-set of size M.

5 CIRCUIT OBFUSCATION BY RESYNTHESIS

As we already mentioned in Section 1, there are two tangled goals in logic encryption: locking and obfuscation. Locking is a logical request to make sure that the *almost correct* behavior cannot happen when a wrong key is applied and an almost correct circuit cannot be easily figured out by studying the *logic* of the encryption circuit. Obfuscation is a structural request to make sure that the correct circuit cannot be revealed by any *structural* analysis of the encryption circuit.

The theory we developed in Sections 3 and 4 achieves the goal of locking without being bothered by obfuscation. In other words, the encryption circuit we designed in these sections are *just an equivalent function of the obfuscated circuit that will be ultimately used.* Obviously, the design in Figure 4 has many vulnerabilities in terms of structural analysis attacks, including subcircuit identification and removal, or general learning. For example, the secret constant k^* can be easily discovered in the circuit structure. Even this is hidden, identifying $f(x)$ XORing other part of the circuit also makes it vulnerable. Therefore, circuit obfuscation is necessary to hide the secret k^* and vulnerable structure of the logic encryption.

Actually, we can prove that any logic encryption is logically almost equivalent to the general scheme in Figure 2. The more precise statement is given in the following lemma. Even though we cannot prove that every encryption should have the XOR of x and k, as discussed in Section 3, it may be the best choice for achieving the linear-size encryption.

LEMMA 5.1. *Any logic encryption $g(x, k)$ for any function $f(x)$ is logically equivalent to the circuit shown in Figure 5.*

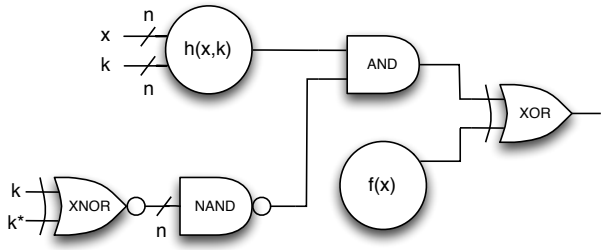

Figure 5: Every possible logic encryption is equivalent to this circuit.

PROOF. Based on the theory in Section 3, the proof is simple. For any logic encryption $g(x, k)$, we can do the Shannon decomposition on x. We then study the structure of $g(x, k)$ based on whether $g(x, k) \neq f(x)$, which gives us the XORing of $f(x)$ with the remaining of the circuit. Since $g(x, k^*) = f(x)$, we have to make the subcircuit XORing with

$f(x)$ being zero if $k = k^*$, which gives the ANDing of the $k \neq k^*$ part. Now, the remaining part is identified as $h(x, k)$ shown in the figure. □

Shortly after Barak et al. [2], Goldwasser and Rothblum [13] had proposed an alternative definition of obfuscation: *best-possible obfuscation*. Intuitively, a best-possible obfuscation of a circuit can be as leaky as any circuit that is equivalent to the given circuit. They also proved that any best-possible obfuscater is also an indistinguishability obfuscater, and for efficient obfuscation, they are equivalent. For logic encryption, it is required that the obfuscation is still an efficient combinational circuit. Thus, an indistinguishability obfuscater is also a best-possible obfuscater.

Li and Zhou [14] has developed a sequential circuit locking and obfuscation technique based on the best-possible obfuscation. He applied structural transformation including retiming, resynthesis, sweep, and conditional stuttering to lock and then obfuscate a given sequential circuit. He also proved that those operations are complete for best-possible obfuscation.

In logic encryption, the obfuscation starts with a combinational circuit and ends up with another combinational circuit. Therefore, we can adopt part of Zhou's approach in the following lemma.

LEMMA 5.2. *For (combinational) logic encryption, a best-possible obfuscation can be done by a sequence of resynthesis.*

Of course, the lemma only provides a feasibility guarantee of obfuscation by resynthesis. Specific transformations are needed in order to hide sensitive information and vulnerable structures.

One of the vulnerabilities in our general encryption scheme in Figure 2 is the key checking $k = k^*$ and then disabling the flip of $f(x)$. It has the same functionality and structure as the "login" process of many software systems. Embedded in this structure is a point-function, a function that will produce zero only for one point in the domain. Obfuscation of point-function has been thoroughly investigated in cryptography research [4–6, 16, 26]. Actually, a point-function is used in the proof of the impossibility result of Barak et al. [2].

The basic idea of point-function obfuscation is to use a random oracle O to produce the output $O(k)$ on input k and to compare it with the stored $O(k^*)$. The security is guaranteed by the impossibility to get k^* from knowing $O(k^*)$. In practice, the random oracle is usually substituted by a cryptographic hash function such as MD5 or SHA. Therefore, we suggest to deploy a one-way hash function in our logic encryption scheme. If we want to combine it with the one-way function insertion discussed in Section 4, we have to make it more difficult.

We also want to mix up the logic $f(x)$ with other part of the encryption, especially with k. Thus, we connect the output of the key checking $O(k) \neq O(k^*)$ to XOR gates with a randomly selected bits of x before they feed into f. Figure 6 shows the suggested circuit before further resynthesis operations are applied, where the gray XOR indicates that only a random subset of bits are modified.

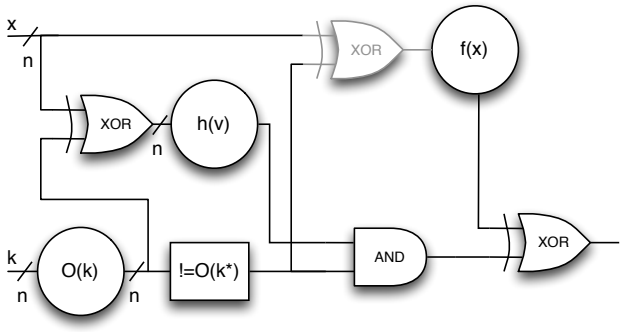

Figure 6: Suggested obfuscation for general logic encryption before further resynthesis; O(k) is one-way hash function, h(v) has an on-set of size M, and gray XOR gate only applies to a small subset of x bits.

6 SCIENTIFIC BENCHMARKS FOR APPROXIMATE ATTACKS

One big challenge for studying approximate attack methods is the lack of scientific measure of their performance. Different from the exact attacks, where the correctness can be easily measured by comparing the keys or by comparing the circuits if there are more than one correct keys, the performances of approximate attacks cannot be easily measured by the generated keys or even the generated circuits. For two approximate attacks, one is better than the other if the circuit generated by one has less error rate than that generated by the other. Exact measure of error rate needs to do circuit simulation for all possible inputs or to do SAT to find the all errors one by one. None of them is cheap.

Not any better is the current practice of using the combination of a traditional encryption and a specific encryption against the SAT-based attack such as Anti-SAT [27]. The easy thing to report is the number of benchmarks where the key to the traditional encryption is correct. However, if an approximate attack could not get the key to the traditional encryption correct, which is very common for large or complex benchmarks, we get lost again. Measuring how many bits are correct in the key to the traditional encryption is of no use, since a mistake on one bit may have more errors than a mistake on many other bits. Using random sampling for error rate measurement is relatively cheap but its accuracy is highly in doubt.

For these reasons, it is desperate for the study of approximate attacks to develop a set of scientific benchmarks to measure their performances. The desired properties for the scientific benchmarks include:

(1) Different keys should have different error rates;
(2) The error rate is known for each key;
(3) The error rates should be adjustable;
(4) The benchmarks should be difficult for the SAT-based attack.

Based on the theory developed in Section 3, we are able to design such a suite of scientific benchmarks. The general design is given in Figure 7.

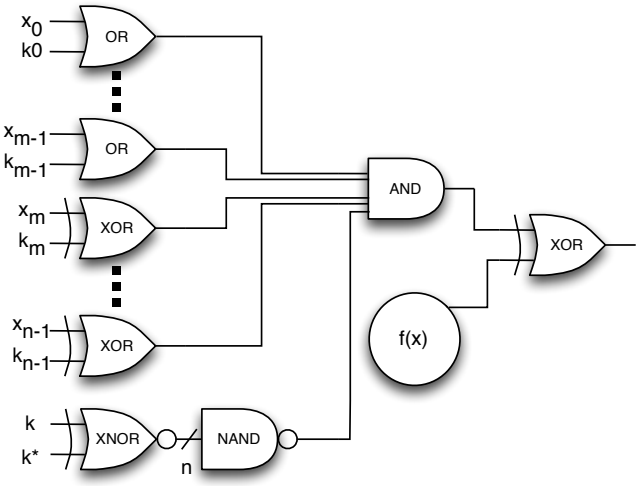

Figure 7: The general design scheme for our scientific benchmarks

Here we assume that the original circuit $f(x)$ has n input bits and the logic encryption has n key bits. We are going to select a correct key k^*, and a number $m < n$ as a design parameter. As shown in the figure, we are going to have an OR of x_i and k_i for all $i \in 0..m-1$ and to have a XOR of x_i and k_i for all $i \in m..n-1$. Then we feed all these n output to an AND gate. The equivalence of k and k^* will generate a zero also to the same AND gate. Then the output of the AND gate is used to flip the output of $f(x)$ by XOR. Here, we do not worry about circuit analysis attacks to these benchmarks, since we can do resynthesis to hide the circuit structure.

We can also introduce randomness to the benchmarks by randomly selecting k^*, randomly selecting the m indices for the OR gates (the remaining will be XOR gates), and randomly inserting an inverter after each key bit in front of the circuit. A similar but more complex benchmark is given in Figure 8. Now we can prove the following theorem for our scientific benchmarks.

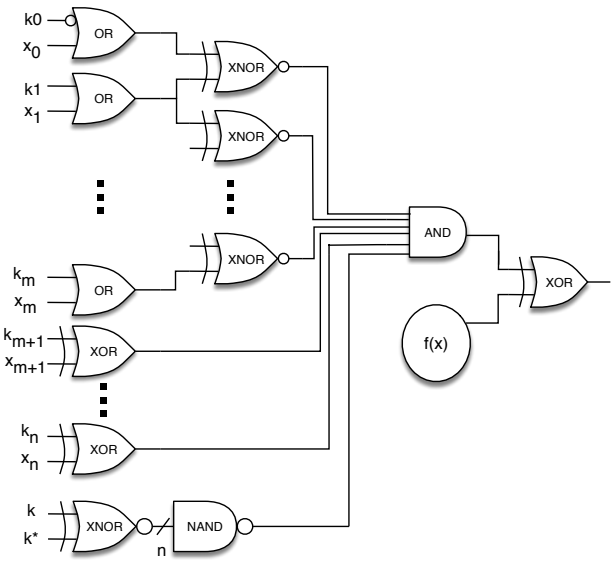

Figure 8: Scientific benchmarks with extra XNOR gates

THEOREM 6.1. *The scientific benchmarks designed as shown in Figure 7 will have different error rates ranging from 2^{-n} to 2^{m-n}. The error rate is known for each key, and the minimal number of iterations for the SAT-based attack is 2^{n-m}.*

The upper bound of error rate happens when k_i for every $i \in 0..m-1$ is set to one, so the value of flip signal depends on the result of XOR gate. For a random assignment of k_i for all $i \in m..n-1$, the flip signal is 1 only if x_i is the opposite of k_i for all $i \in m..n-1$. So the error rate is $2^m/2^n = 2^{m-n}$.

The lower bound of error rate happens when k_i for every $i \in 0..m-1$ is set to zero, so the flip signal is one only if $x_i = 1$ for $i \in 0..m-1$ and $x_i = \bar{k_i}$ for $i \in m..n-1$. In that case, the error rate is 2^{-n}.

Any other key values will have error rate ranging from 2^{-n} to 2^{m-n}. To solve the correct key, the SAT-based attack should prune out all wrong keys, so the number of iterations for the SAT-based attack is at least 2^{n-m}.

7 EXPERIMENTAL RESULTS

Based on the theory developed in the paper, we can design the logic encryption for any circuit to fully determine its attack complexity (in terms of the number of queries to the original circuit) and the error rate for wrong keys. In this section, we conduct experiments on the SAT-based attack [24] to verify our theory, to check the effectiveness of Goldreich's one-way function [12], and to measure the actual attack time by the SAT-based attack.

It should be noted that in our encryption, the attack complexity and error rate is independent of the original circuit

$f(x)$. We have verified this by checking the attack time by SAT-based attack on the same encryption on a set of different original circuits. In fact, even using a constant function $f(x) = 0$ gives us the same result. Therefore, in our later reports, we will not going to specify what is the original circuit.

We use the simple logic encryption design shown in Figure 3. Please recall that in this encryption, the $h(v)$ function has an on-set of $2^{n/2}$. Thus, it has $2^{n/2}$ as both attack complexity and error number. We have created a sequence of encryptions with the input lengths ranging from 12 to 26. Then we run the SAT-based attack on them and collect the runtime. The result is plotted in Figure 9, where the x-axis shows the input lengths and y-axis gives the attack time in log scale.

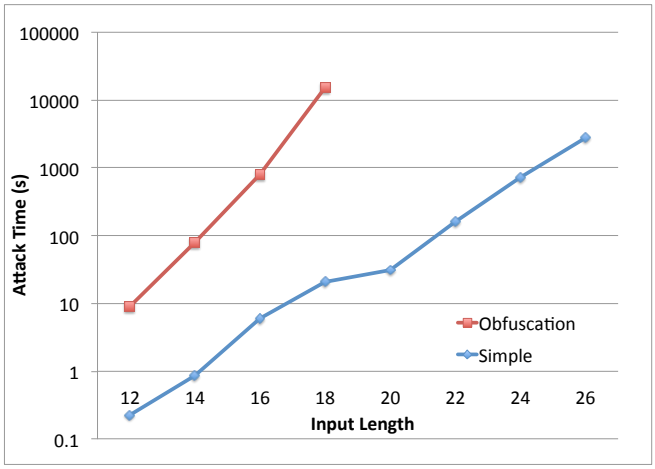

Figure 9: Runtime by SAT-based attack on simple encryptions in Figure 3 and obfuscation in Figure 6.

We also want to check the effectiveness of using Goldreich's one-way function in logic encryption. We implement the first step of obfuscation shown in Figure 6 on top of the simple logic encryption in the first set of experiments. The $O(k)$ function in the circuit is implemented by Goldreich's one-way function with an input length of 80 and an output length of 40. We use the simple $P = x_0 \oplus x_1 \oplus x_2 \oplus (x_3 \wedge x_4)$ discussed in Section 4. To minimize the disturbance on the attack complexity, the gray XOR function is not implemented. The SAT-based attack is run on these encryptions with different input bit-length and the results are shown in Figure 9.

As can be verified in Figure 9, both the simple logic encryption and its obfuscation with Goldreich's one-way function have exponential growths of attack time in terms of the input lengths. With Goldreich's one-way function, the growth of the attack time becomes bigger and steeper. They confirms our theory for logic encryption in the paper.

8 CONCLUSION

We have developed a theory for logic encryption in the paper. Our development started with a seperation of the two entangled goals in logic encryption, that is, a logic requirement of locking and a structural requirement of hiding. We consider only the logic locking in the first part of the paper, and developed a theory that gives a complete view of the logic encryption design space, and its relations with attack complexity and error rate. In the theory, we also proved a contention between attack complexity and error rate. A general logic encryption circuit of linear size is derived from the theory.

Circuit obfuscation is applied on top of our logic encryptions to address the structural requirement of hiding. We discussed the current development in cryptographic obfuscation, and showed that resynthesis operations are complete for best-possible obfuscation. We also discussed approaches to use one-way functions to protect the sensitive key checking process and to burden the SAT engine in attack.

We tested our logic encryption designs and the obfuscation with one-way function by the SAT-based attack. The experimental results have confirmed our theory and the robustness of our approach.

REFERENCES

[1] N. Alon. 1990. Transversal numbers of uniform hypergraphs. *Graphs Combin.* 6 (1990), 1–4.

[2] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. 2001. On the (Im)possibility of Obfuscating Programs. In *International Cryptology Conference*. 1–18.

[3] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. 2010. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design and Test* 27, 1 (2010).

[4] R. Canetti. 1997. Towards realizing random oracles: Hash functions that hide all partial information. In *International Cryptology Conference (LNCS 1294)*, B. S. Kaliski Jr. (Ed.). 455–469.

[5] R. Canetti and R. R. Dakdouk. 2008. Obfuscating point functions with multibit output. In *IACR Conference on the Theory and Applications of Cryptographic Techniques (LNCS 4965)*, N. P. Smart (Ed.). 489–508.

[6] R. Canetti, Y. T. Kalai, M. Varia, and D. Wichs. 2010. On symmetric encryption and point obfuscation. In *IACR Theory of Cryptography Conference (LNCS 5978)*, D. Micciancio (Ed.). 52–71.

[7] V. Chvátal and C. McDiarmid. 1992. Small transversals in hypergraphs. *Combinatorica* 12 (1992), 19–26.

[8] J. Cook, O. Etesami, R. Miller, and L. Trevisan. 2009. Goldreich's One-Way Function Candidate and Myopic Backtracking Algorithms. In *IACR Theory of Cryptography Conference (LNCS 5444)*, O. Reingold (Ed.). 521–538.

[9] O. Coudert. 1996. On Solving Covering Problems. In *Proc. of the Design Automation Conf.*

[10] Sophie Dupuis, Papa-Sidi Ba, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre. 2014. A Novel Hardware Logic Encryption Technique for Thwarting Illegal Overproduction and Hardware Trojans. In *IEEE International On-Line Testing Symposium*. 49–54.

[11] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. 2013. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *Proc. IEEE Symposium on the Foundations of Computer Science*. 40–49.

[12] Oded Goldreich. 2000. Candidate One-Way Functions Based on Expander Graphs. *Electronic Colloquium on Computational Complexity* 7, 90 (2000).

[13] Shafi Goldwasser and Guy N. Rothblum. 2007. On best-possible obfuscation. In *Proceedings of the 4th conference on Theory of cryptography*. 194–213.

[14] Li Li and Hai Zhou. 2013. Structural Transformation for Best-Possible Obfuscation of Sequential Circuits. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*. 55–60.

[15] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Pan. 2016. Provably Secure Camouflaging Strategy for IC Protection. In *Proc. Intl. Conf. on Computer-Aided Design*. Austin, TX.

[16] B. Lynn, M. Prabhakaran, and A. Sahai. 2004. Positive Results and Techniques for Obfuscation. In *IACR Conference on the Theory and Applications of Cryptographic Techniques*.

[17] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. 2012. Logic encryption: A fault analysis perspective. In *Proc. DATE: Design Automation and Test in Europe*. 953–958.

[18] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. 2013. Security analysis of integrated circuit camouflaging. In *CCS*.

[19] Jeyavijayan Rajendran, Huan Zhang, Chi Zhang, Garrett S. Rose, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. 2015. Fault Analysis-Based Logic Encryption. *IEEE Trans. Comput.* 64, 2 (2015).

[20] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *Proc. DATE: Design Automation and Test in Europe*.

[21] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. 2017. AppSAT: Approximately Deobfuscating Integrated Circuits. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*. 95–100.

[22] Kaveh Shamsi, Travis Meade, Meng Li, David Z. Pan, and Yier Jin. 2019. On the Approximation Resiliency of Logic Locking and IC Camouflaging Schemes. *IEEE Trans. Information Forensics and Security* 14, 2 (2019), 347–359.

[23] Yuanqi Shen and Hai Zhou. 2017. Double DIP: Re-Evaluating Security of Logic Encryption Algorithms. In *Proc. ACM Great Lakes Symposium on VLSI*. 179–184.

[24] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the Security of Logic Encryption Algorithms. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*. 137–143.

[25] L. G. Valiant. 1984. A Theory of The Learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.

[26] H. Wee. 2005. On obfuscating point functions. In *Proc. ACM Symposium on the Theory of Computing*, H. N. Gabow and R. Fagin (Eds.). 523âĂŞ532.

[27] Yang Xie and Ankur Srivastava. 2016. Mitigating SAT Attack on Logic Locking. In *Conference on Cryptographic Hardware and Embedded Systems*. 127–146.

[28] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan J V Rajendran, and Ozgur Sinanoglu. 2016. SARLock: SAT attack resistant logic locking. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*. 236–241.

[29] M. Yasin, J. Rajendran, O. Sinanoglu, and R. Karri. 2016. On improving the security of logic locking. *IEEE Transactions on Computer Aided Design* 35, 9 (Sept. 2016).