Resolving the Trilemma in Logic Encryption

Hai Zhou, Amin Rezaei, and Yuanqi Shen
Northwestern University

Abstract—**Logic encryption, a method to lock a circuit from unauthorized use unless the correct key is provided, is the most important technique in hardware IP protection. However, with the discovery of the SAT attack, all traditional logic encryption algorithms are broken. New algorithms after the SAT attack are all vulnerable to structural analysis unless a provable obfuscation is applied to the locked circuit. But there is no provable logic obfuscation available, in spite of some vague resorting to logic resynthesis.**

In this paper, we formulate and discuss a trilemma in logic encryption among locking robustness, structural security, and encryption efficiency, showing that pre-SAT approaches achieve only structural security and encryption efficiency, and post-SAT approaches achieve only locking robustness and encryption efficiency. There is also a dilemma between query complexity and error number in locking. We first develop a theory and solution to the dilemma in locking between query complexity and error number. Then, we provide a provable obfuscation solution to the dilemma between structural security and locking robustness. We finally present and discuss some results towards the resolution of the trilemma in logic encryption.

1. Introduction

Logic encryption (aka logic locking or logic obfuscation) has been proposed over many years, as an effective IP protection technique that modifies a given logic circuit with the introduction of a set of key inputs. Traditional approaches to logic encryption [1, 18, 5, 14, 15, 10, 25, 16] were all based on ad-hoc approaches to select a subset of internal signals in the original circuit to be modified by key-bits. An example is shown in Figure 1. Please observe that in such approaches, the original circuit structure had not been hidden from the attacker, and the protection depends mainly on the attacker's assumed incapability to figure out the correct polarities on the selected signals.

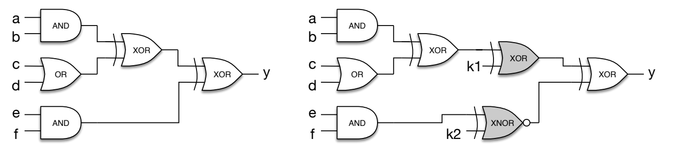

Fig. 1. Traditional logic encryption example: gray gates are lock gates.

However, such assumptions had been shown to be invalid with the success of the SAT-based attacks on almost all of the traditional approaches [23]. The main idea of the SAT-based attack is to use a "miter" circuit (two copies of the encryption circuit with the same primary inputs and different primary output) to identify a primary input value that can still differentiate the two circuits. Then the correct input-output relation from a query to the functioning circuit is used to add a constraint on both key inputs in the "miter". Its effect is to remove the wrong keys with effect on that input. Only correct keys are remaining in the constraints if the two copies cannot be differentiated. The pseudo-code of the SAT-based attack is given in Algorithm 1. **The power of the SAT-based attack lies on the fact that many a query removes a large number of wrong keys.**

After the SAT-based attack, many different defenses have been proposed [29, 28, 26, 32, 31, 20, 30, 13, 17, 19, 27]. A

Algorithm 1 SAT-Based Attack

```
Input: An encryption circuit $g(x,k)$ and the oracle access to the target $f(x)$.
Output: 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:     $\hat{y} = f(\hat{x})$;
3:     $g(x,k) = g(x,k) \wedge (g(\hat{x},k) = \hat{y})$;
4:     $g(x,k1) = g(x,k1) \wedge (g(\hat{x},k1) = \hat{y})$;
5: end while
6: $k^* = SAT(g(x,k))$;
```

contrasting feature of all these post SAT-attack solutions from the traditional solutions is that, the logic encryption now shifts its attention to the design of the *difference logic.* The difference logic is defined as the difference (XOR) of the encryption circuit and the original circuit, and thus is a Boolean function of primary inputs and key inputs. Different from traditional solutions (where the difference logic is a by-product of internal signal modifications and thus hard to know), a well-designed difference logic can guarantee that an exponential number of inputs are needed in order to remove all wrong keys.

However, making the difference logic explicit introduces a structural vulnerability in the design; the difference logic has to be XORed with the original circuit to form the encryption circuit. Therefore, without structural obfuscation, the original circuit lays exposed for piracy. This can be easily seen in Figures 4 and 5 in the next section. However, obfuscation, as a technique against structural analyses, is very hard, since it is impossible to exhaust structural analyses. Even though we can show that resynthesis is a viable way to obfuscation, conventional synthesis tools were mainly engineered for optimization and thus are not suitable for this purpose.

We can identify three requirements in logic encryption: locking robustness, structural security, and encryption efficiency. The *locking robustness* is measured by the number of required queries to the functioning circuit (oracle) in order for find the correct key. The *structural security* is measured by the required time to find out the target circuit through structural analysis of the encryption circuit. The *encryption efficiency* is given by the length of the key and the overhead of the size and delay in the encryption circuit.

There is a trilemma in logic encryption among locking robustness, structural security, and encryption efficiency. All existing solutions have achieved at most two of these requirements. For example, all pre-SAT approaches have achieved structural security and encryption efficiency, through direct locking of internal signals in the target circuit. They are structurally secure since no vulnerable locking structure is introduced, and efficient because the key length and overhead are small. However, SAT attack demonstrated that they are not robust in locking. On the other hand, all post-SAT approaches have to explicitly design the locking logic via the difference logic. With an efficient explicit difference logic, both locking robustness and encryption efficiency are achieved. But, the logic encryption given by XORing the original circuit with the difference logic is unavoidably vulnerable by structural analysis.

Most post-SAT solutions, especially the earlier ones including SARLock [28], Anti-SAT [26], and TTLock [30], have very low error rate for any wrong key. These designs are vulnerable to approximate attacks such as Double-DIP [22] and AppSAT [21], or the bypass attack []. We will first examine the dilemma beween query complexity and error number, and provide solutions achieving the best trade-off between them.

All post-SAT solutions are structurally vulnerable and request obfuscation for protection. However, there is no viable solutions to obfuscation in spite of some vague resorting to logic resynthesis. Here, we investigate the dilemma between structural security and locking robustness, and develop for the first time a solution to the logic obfuscation, based on the concept of universal circuit. We acknowledge that our solution achieves structural security and locking robustness, but lacks encryption efficiency.

We have illustrated the trilemma of logic encryption in Figure 2, with both existing and proposed solutions placed in corresponding positions. In summary, we develop the theory and solutions to the dilemma of query-error trade-off in logic locking. Based on it, we also develop the obfuscation theory and solutions to the dilemma of structural security and locking robustness. We also provide some partial results towards the holy grail of logic encryption, achieving all three requirements in the trilemma.

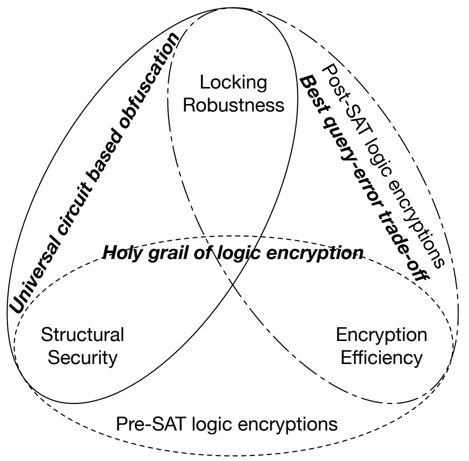
Locking Robustness
Universal circuit based obfuscation
Post-SAT logic encryptions
Best query-error trade-off
Holy grail of logic encryption
Structural Security
Encryption Efficiency
Pre-SAT logic encryptions

Fig. 2. The trilemma among locking robustness, structural security, and encryption efficiency in logic encryption, and the situations of existing and proposed (in bold italic) solutions.

The remaining of the paper is organized as follows. In Section 2, the attack model will be formally defined and discussed. In Section 3, the logic locking subproblem will be studied without considering structural security. A theory showing the dilemma between query complexity and error number will be developed, and efficient lockings with the best trade-off between query complexity and error number will be designed. In Section 4, Our focus will then be shifted to structural obfuscation, the harder subproblem of logic encryption. The difficulty mainly comes from the fact that it is impossible to exhaust all possible structural analyses on a circuit. We will first formulate the problem as *indistinguishability Logic Obfuscation (iLO)*, and then provide provable solutions based on the concept of universal circuits [24, 8].

With a symptomatic size of $O(n \log n)$, the universal circuit based solutions might be considered efficient in theory. However, for practical deployment, our criterio for encryption efficiency must be higher. The holy grail of logic encryption should target a size overhead below the original circuit size and a key length at most twice of the input length. We will present our efforts in pursuing the holy grail in Section 5.

2. Attack Model and the Trilemma of Logic Encryption

In this section, we are going to review the brief history of logic encryption, discuss various scenarios and implied requests in hardware IP protection, then define the corresponding general attack models, and formulate the related IP protection problems. It is our contribution to identify logic locking and structural obfuscation as two separate subgoals of logic encryption. Through a thorough discussion on different IP protection scenarios, we plan to re-define logic locking, logic obfuscation, and logic encryption as subtly different problems in IP protection. Please note these terms are currently used interchangeably in the existing research literature.

2.A. IP Protection Scenarios and Requests

Our paper will start by re-examining different scenarios in hardware IP protection. We emphasize that IP protection is not just one problem but a range of various problems. This can be demonstrated by comparing the **IP protection against a novice** and **that against an expert.**

The former scenario has been the one targeted by the traditional logic encryption. As discussed in the previous section, the traditional approaches had all selected a subset of internal signals in the original circuit to be modified by the key bits. Since most parts of the original circuit were exposed except the polarities of the selected signals, the attacker must be assumed to know nothing about the circuit, not even its functionality. It is equivalent to say that the attacker is a novice.

Now consider the scenario where the attacker is an expert, for example, as a competitor of the target hardware. In this case, the attacker may already know the functionality of the target circuit, may even have a design that is slightly inferiors to the target design. Therefore, the exposure of most parts of the original circuit has already leaked invaluable design information to the attacker. Furthermore, since the attacker already knows the functionality, there is no need to query the functioning circuit.

Based on the discussion, it can be claimed that not only the traditional logic encryption but also all the logic encryption after the SAT-based attack are for IP protection against a novice.

2.B. Attack Models and IP Protection Problems

The paper will study a variation of different attack models. In addition to different prior knowledge and target information as discussed in previous section, an attack model will also include the limit of the attacker's capability in action. Our preliminary study has identified logic analysis and structural analysis as two important aspects in the attacks on logic encryption. However, nothing can forbidden an attacker from combining logic and structural analyses in sophisticated ways.

We want to illustrate such a sophisticated way of attack with an example of a very new logic encryption approach called SFLL (Stripped Function Logic Lock) [30]. SFLL is generally viewed by the community as the most advanced and thus perhaps the most secure approach to logic encryption (or locking). Recall that approaches after the SAT-based attack have to design the difference logic and then XOR it with the original circuit. The usual way to ensure that the difference logic not only produces errors but also has a correct key is to mask one key value from producing errors. SFLL has moved one step forward by moving the mask from the difference logic into the original circuit. In other words, it first does the same difference on the original circuit but with the keys fixed at specific values, then the difference logic is applied (i.e. XORed) to the modified circuit (named SF). The comparison between SFLL and other post-SAT logic encryption is shown in Figure 3. Even though it still requests a resynthesis to obfuscate the SF circuit, a benefit of SFLL is that the outer difference logic can be left visible without worrying its removal.

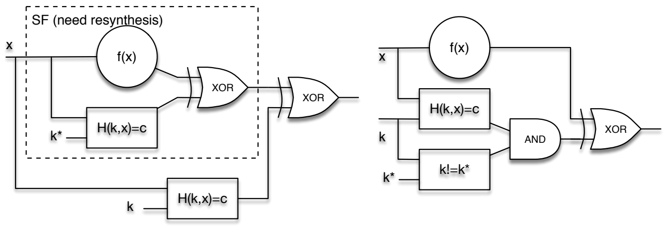

Fig. 3. Comparison between SFLL and other post-SAT logic encryptions.

A main difference logic proposed in SFLL is the constant Hamming distance function, that is, a difference will be produced if the distance of the key k and the input x is the given constant, denoted as $H(k,x) = c$. Our attack assumes that one of the modified inputs $\hat{x}$ has been encountered (which is a reasonable assumption since a locking never producing any error is tantamount to no locking). It means that we know $H(k^*, \hat{x}) = c$. **Now there is a linear time algorithm with queries to the functioning circuit to find k^*.** Denote by $\hat{x}[i,j]$ the same $\hat{x}$ except that the ith and jth bits are flipped. Querying the functioning circuit on $\hat{x}[0,i]$ for $i = 1$ to $n-1$ and comparing with the results on SF circuit will tell us whether $H(k^*, \hat{x}[0,i]) = c$. It will partition the bit indices into two groups, with index 0 in the group satisfying the equation. It can be shown that at least one group has exactly c members. Denoting this group by C, we can show that $\hat{x}$ with all the bits indexed by C flipped must be k^*. **It is important to observe here that the queries to the functioning circuit are well structured, and the results are used in organized ways with structural analysis, much differently from the SAT-based attack.**

Our paper will study the novice attacker model at one end of the spectrum, where the attacker is assumed to know nothing about the protected circuit and its target is to find out the correct key. In the scope of attack capabilities, we plan to consider logic analysis, structural analysis, and any combinations of them. Notice that an attack model with only logic analysis is no longer valid after the SAT-based attack. All attack models considered in this paper will include structural analysis. However, capturing the capability of structural analysis and especially that of combined structural and logic analyses will be one of the biggest challenges to be overcome.

3. Theory and Solution to Query-Error Dilemma

3.A. Contention Between Query Complexity and Error

Since each iteration of the SAT-based attack is to find an input that can differentiate two keys on $g(x,k)$ and then to constrain the keys by the evaluation of that input on the original circuit $f(x)$, we start our investigation by a Shannon decomposition of the encryption circuit $g(x,k)$ on the input x. It gives us the following equation:

$$g(x,k) = \bigvee_{i=0}^{2^n-1} m_i(x) \wedge g(i,k),$$

where $m_i(x)$ is the ith minterm of x, and $g(i,k)$ is Boolean function of only k.

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 [28, 26] 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

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

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

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

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^m-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 [2] 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 [6] have given another upper bound of $(2^n + \lfloor M/2 \rfloor 2^m)/\lfloor 3M/2 \rfloor$. 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 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

$$2^m(1-M/2^n)^N \le 2^m e^{-m} = (2/e)^m.$$

□

3.B. The Best Query-Error Trade-Off

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 [2, 6].

Lemma 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 [2] and Chvátal and McDiarmid [6].

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 4, which is also given by the following formula,

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

Its correctness is stated in the following theorem.

Theorem 3: If function $h: B^n\to B$ has an on-set of size M, then the logic encryption given in Figure 4 will have an error number of M for every wrong key, and a minimal attack complexity at least $2^n/M$.

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 5. 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.

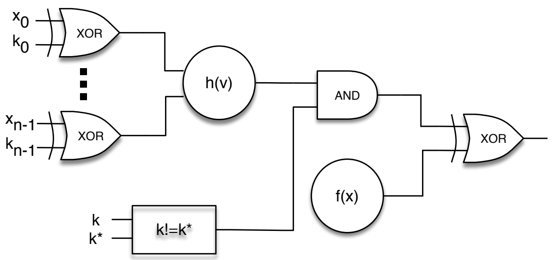

Fig. 4. A general logic encryption scheme where h(v) has an on-set of size M.

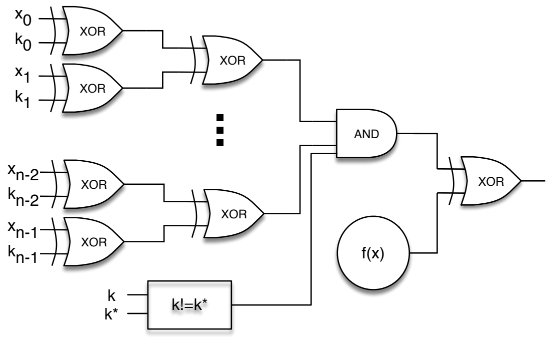

Fig. 5. A simple logic encryption design with both exponential attack complexity and error number.

3.C. 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 [26]. 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.

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. [7] 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.

4. Theory and Solution to Locking-Obfuscation Dilemma

This section will investigate the logic obfuscation problem, that is, how to protect a circuit from structural analysis. Logic obfuscation is needed in at least two scenarios. First, in IP protection against an expert attack, obfuscation is needed for protection of target information. Second, in post-SAT logic encryptions, obfuscation is needed to protect the lock.

4.A. Indistinguishability Logic Obfuscation

Obfuscation is a technique to transform one implementation (a circuit or a program) into another such that the sensitive information in the former is protected. The difficulty of obfuscation comes from the fact that, it is hard to identify the target information that needs to be protected, and it is even harder to show that it is actually protected in the transformation. It is so because it is extremely difficult to capture what can be done by the attacker in the analysis of the obfuscated implementation.

In logic encryption, we separate attack analysis into logic analysis and structural analysis. The former is defined as the analysis using only logic information but none of the structural information, with the SAT-based attack as a typical example. The latter is defined as the analysis that has to use the structural information, with removal attack and signal activity analysis as examples. It must be reminded that based on the definition, structural analysis could use logic information and queries to the functioning circuit. Therefore, complicated analyses combining structural and logic information in sophisticated ways (such as the attack on SFLL shown in Section 2.B) also belong to structural analysis by definition.

In order to escape from the doom of cat-and-mouse chasing between attacks and defenses, it is very important for us to define an attack model that is as general as possible to capture all possible structural analyses as discussed. It seems to be an insurmountable task. Fortunately, we have already a model to follow in cryptography and theoretical computer science.

Barak et al. [4] was the first to take the Herculean task to investigate the cryptographic (program) obfuscation. There, an obfuscation is defined as a random transformation conducted on a given program, that keeps the functionality and performance (with a possible polynomial time slow-down) but hides the original program. The way they used to capture the seemingly inexhaustible attacker behavior is to model it as any probabilistic polynomial time algorithm. To show that a given information is hidden by the obfuscation from the attacker, it is sufficient to show that there exists a simulator without using that information whose behavior is indistinguishable from the attacker algorithm.

The most powerful obfuscation is the one that hides all information in the program, also known as "black-box" obfuscation. An important result in Barak et al. [4] is the proof that the black-box obfuscation does NOT exist for general programs or circuits. With that, it also proposed a weaker form, called *indistinguishability obfuscation (iO)*. An iO is defined as such an obfuscation that, when applied to any two programs of the same function, produces results that are computationally indistinguishable from each other.

A big breakthrough along this line of research was the discovery of a candidate of iO in 2013 by Garg et al. [11]. Even though the candidate has provable promising cryptographic properties, its construction is currently still very expensive. Apon et al. [3] had investigated the implementation of the candidate, and found that the obfuscation was "still far from being deployable, with the most complex functionality we are able to obfuscate being a 16-bit point function." Given that their implementation was a program while logic obfuscation requires a combinational circuit, what we can expect is that the iO is even more expensive to be implemented in our paper.

Fortunately, we have just discovered that our logic obfuscation problem is different from the cryptographic (program) obfuscation, and luckily, our problem is easier than the cryptographic obfuscation. The critical difference is that the logic obfuscation has key inputs while the cryptographic obfuscation does not. In fact, the request to have key inputs is an intrinsic feature of hardware IP protection. If we have an obfuscated circuit still functioning as the original circuit, as in the case of cryptographic obfuscation, then an attacker can simply copy it without bothering to analyze it. As will be shown in the next section, when we allow key inputs and have protection mechanism for them, the logic obfuscation problem becomes simpler than the cryptographic obfuscation.

In this paper, we plan to develop a solid foundation for the logic obfuscation, following the model established in iO [4, 11]. **The *indistinguishability Logic Obfuscation (iLO)* will be defined as the logic obfuscation with key inputs that, when applied to any two circuits of the same function, produces results that are computationally indistinguishable from each other.** Under this formulation, an attacker's any possible analysis on the obfuscated circuit, modelled as a probabilistic polynomial time algorithms, cannot distinguish it from the obfuscation of any other equivalent circuit. Therefore, all structural information is protected.

4.B. Solutions Based on Universal Circuits

As already pointed out in the introduction, logic obfuscation is the most important and most challenging part of hardware IP protection. **The novelty and success of our paper will be built on top of our recent discovery of promising candidate solutions to the solidly defined indistinguishability logic obfuscation (iLO).**

The candidate solution is based on the concept of universal circuit. A *universal circuit* is a circuit that can implement any function in a given family by fixing the values on some selected inputs (leaving the other inputs as the primary inputs for the target function). Valiant [24] has developed a universal circuit that can implement any circuit whose number of inputs and gates is upper bounded by n. He provided two constructions using so-called 2-way and 4-way constructions, with sizes $5n \log n$ and $4.75n \log n$, respectively. Cook and Hoover [8] have developed a family of depth-universal circuits. For any n, c, d they gave a universal circuit with depth $O(d)$ and size $O(c^3 d / \log c)$ that can implement any circuit having n inputs, of size c and depth d.

Our discovery can be summarized in the following theorem.

Theorem 4: A universal circuit for any family of circuits is an indistinguishability logic obfuscation (iLO) for that family.

These candidate solutions to iLO is much more efficient than any candidate solution to the iO, where a universal circuit is only one step in the long process. This is because key inputs are allowed and assumed to be protected in iLO. The prevailing technology for key protection in existing logic encryption is tamper-proof register, which is not cheap. **Therefore, our paper will treat the requested key bits in the implementation as important resources, and will develop approaches to minimize the number of key bits too.**

We also plan to develop solutions to the logic encryption problem by combining the solutions to iLO and the solutions developed for logic locking in Section 3. A simple approach could be, first to lock the original circuit as one shown in Figure 4, and then to use a universal circuit to obfuscate the locked circuit. The key bits are composed of the keys bits in locking and those in obfuscation. The existence of extremely efficient logic locking circuit as shown in Figure 5 actually provides us a more efficient alternative. That is, we can simply omit the logic locking step and only do the obfuscation targeting a bit larger circuit size than the original one. It is correct because now the obfuscation is secure for a family already including the locking circuit.

We want to investigate whether the iLO based on the universal circuit is still secure for logic locking even when the targeting circuit size is not increased. Pushing further, we would want to study how far we can reduce the targeting circuit size while still maintaining the validity of solutions. A challenge here is that, when the targeting circuit size is smaller than that of the circuit to protect, the universal circuit cannot guarantee its implementation. Some specific structure of the circuit must be explored for the implementation. Hence, there is a risk that the exploration may leak information about these structures. But we also have to note that not every information is sensitive. For example, Valiant's universal circuit actually gives out the information on circuit size, but there is generally no risk.

For modern circuit implementations, the performance is given a much higher priority than the size. Since the performance of a circuit depends heavily on its depth, we would want to develop iLO solutions based on universal circuits that have shallow depth. Valiant's circuits, even though efficient in sizes, have large depths. For the purpose of performance, Cook and Hoover's circuit may be the better starting point. **We will investigate the size decreasing schemes (as discussed in the previous paragraph) for both Valiant's and Cook and Hoover's circuits.**

5. Resolving the Trilemma of Logic Encryption

The universal circuit based solutions in the previous section have achieved both structural security and locking robustness. The trilemma will be resolved if the efficiency can also be achieved on these solutions.

5.A. Solutions Based on Quasi-Universal Circuits

As shown in the previous section, the universality in the obfuscation circuit helps to hide the structural information. However, it is also the universality that induces a high implementation cost, on circuit performance, size, and key length. But since not every structural information is sensitive in logic encryption, we may not need a truly universal circuit in our implementation. We call a circuit that can implement a sufficient number of circuits but not all circuits a *quasi-universal circuit.* **We want to explore and develop efficient quasi-universal circuits that are still secure for logic obfuscation.**

One direction to explore is to restrict the universality to only signal routing. Recall that Valiant's universal circuit has universality both on signal routing and gate functionality. Here we would rather fix the gate functions in the circuit while introducing flexibility in signal routing. One possible design is to fix all the gates to be AND, but allow the signal polarities as well as their routing to be programable. For that purpose, any candidate connection between two gates will pass through two key gates each with a key bit: an XOR gate and an OR gate. The first one is used to decide the polarity and the second one whether the signal is routed.

We will first establish that if we allow all possible connections in a topological order, then our design is as universal as Valiant's. The design will also have depth of at most $3d$ for any target circuit of depth d. The only problem is that the maximal fanin and fanout degrees could reach $n - 1$.

In order to reduce the fanin and fanout degrees in the above design, we plan to group the gates into different levels and to have connections only between adjacent levels. An important problem here is to decide how many gates are needed in each level. One possible solution is to make the decision based on the target circuit. We can first do a breadth-first traversal of the target circuit (while ignoring inverters) to assign a level to each gate based on its longest path from the primary inputs. Then we allocate at least the same number of gates to each level. Adding more gates in each level can help to fool the attacker and even increase the solution space to include the logic locking discussed in Section 3.

To further reduce the fanin and fanout degrees, we need to avoid full connections between adjacent levels if the number of gates in the levels are too many. We plan to have a design where the levels closer to the primary inputs have more gates than those closer to the primary outputs. A threshold t will be selected, and every gate in each level will be connected to at most t gates in the next level. When the maximal number of gates in adjacent levels is at most t, we have full connections. When the maximal number of gates is much larger than t, we get convolutional connections. We called such a design *a convolutional circuit,* as shown in Figure 6.

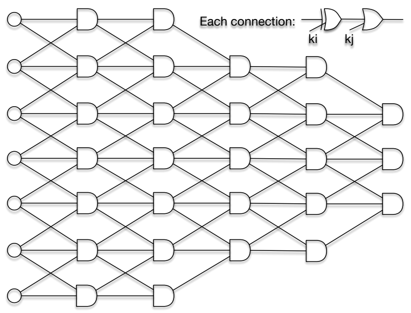

Fig. 6. One example of convolutional circuit with fanout degree of three.

5.B. Solutions Starting with the Original Circuit

As already discussed, if logic locking is designed as a difference logic then XORed with the target circuit, then obfuscation is needed to protect the lock. The approach of first locking and then obfuscating will introduce a lot of keys and overhead. In this section, we will develop combined logic locking and obfuscation in order to get efficient logic encryptions.

In evaluating existing logic encryption algorithms, Subramanyan et al. [23] have found that placing XOR key gates on the primal inputs of an AND-tree gives a hard case for SAT-based attack. We have discovered here is that if the target circuit has very biased on- and off-set, then placing XOR/XNOR key gates on the primal inputs forms an ideal logic locking.

Lemma 5: If a target circuit $f(x)$ has a smaller on- or off-set of size M, then a locking by placing XOR/XNOR key gates on the primal inputs has an average attack complexity of $2^n/M$.

Please note that this locking scheme is different from that in Figure 4. Here, different target circuit will have different M which cannot be selected in locking. However, the benefit is that no obfuscation is needed now since the lock is hidden.

If we have a target circuit with very biased on- and off-set sizes, meaning that $2^n/M$ is exponential in terms of n, then we already have an efficient logic encryption, with combined logic locking and obfuscation. Unfortunately, it does not work if $2^n/M$ is small.

6. Experimental Results

To validate our approaches, we have conducted three sets of experiments to check the locking robustness of our designs. The first set is the locking with the best trade-off between query complexity and error number, as shown in Figure 5. The second set is the obfuscation based on quasi-universal circuits, especially that uses the convolution circuit shown in Figure **??**. The third set is the combined logic locking and obfuscation when the target circuit is very biased on its on- and off-set.

We apply the SAT-based attack [23] to the locking design in Figure 5 and the same design with Goldreich's one-way function [12] applied on the key input, and to measure the actual attack time by the SAT-based attack.

It should be noted that in the locking design, the query 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 locking on a set of different original circuits.

We first test the simple logic locking design shown in Figure 5. Please recall that this locking has $2^{n/2}$ as both query complexity and error number. We have created a sequence of lockings 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 7, where the x-axis shows the input lengths and y-axis gives the attack time in log scale.

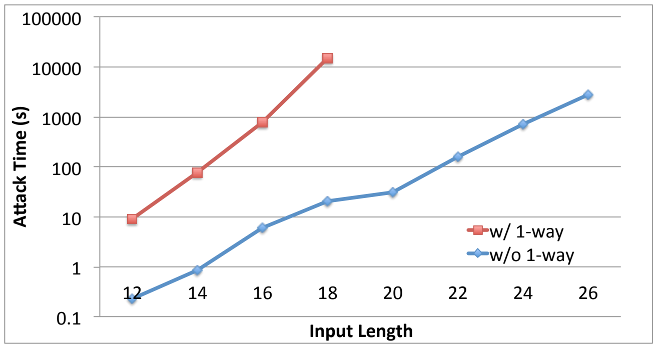

Fig. 7. Runtime by SAT-based attack on lockings without one-way function (Figure 5) and with one-way function.

We also want to check the effectiveness of adding Goldreich's one-way function on the key input. The one-way function we use has an input length of 80 and an output length equal to the circuit input length. We use the simple $P = x_0 \oplus x_1 \oplus x_2 \oplus (x_3 \wedge x_4)$ as discussed in [7]. The SAT-based attack is run on these lockings with different input bit-length and the results are shown in Figure 7. It can be verified that both the simple logic locking and the one with Goldreich's one-way function have exponential growths of attack time in terms of the input lengths.

For the second set of experiments, we have implemented the simple convolutional circuit in Figure 6, with a range of input lengths and depths. On the convolutional circuit, we randomly select the programming bits to fix the target function. We then apply the SAT-based attack to find out the key. In the convolutional circuit, the number of programming bits are huge. In order to reduce the key length, in addition to considering all programming bits as keys ("full key"), we also consider using only half of them as keys ("partial key"). The running time of the SAT-based attacks on them is shown in Figure 8. It can be validated that the attack time is exponential in the input lengths for both cases.

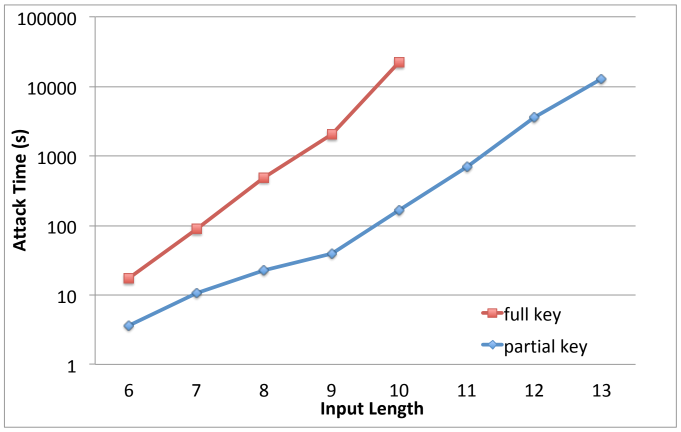

Fig. 8. Runtime by SAT-based attack on convolution circuits with full key and partial key.

The third set of experiments is used to validate the efficient combined locking and obfuscation design for target function with biased on- and off-set. Here, we set up our target function as

$$f(x) = \bigwedge\nolimits_{i \in even} xor(x_i, x_{i+1}).$$

It can be checked that the on-set of the function is $2^{n/2}$, thus is very biased. Based on Lemma 5, the following encryption circuit is good.

$$g(k, x) = \bigwedge\nolimits_{i \in even} xor(xor(k_i, x_i), xor(k_{i+1}, x_{i+1})).$$

The running time of the SAT-based attacks on such an encryption with different input lengths is shown in Figure 9. It can be validated that the runtime is exponential in the length of the inputs.

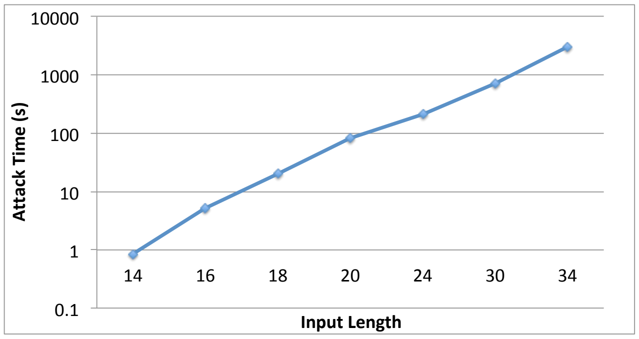

Fig. 9. Runtime by SAT-based attack on efficient locking and obfuscation of biased circuits.

7. Conclusion

Logic encryption is an important technique for hardware IP protection. It has been discovered that there is a trilemma among the locking robustness, structural security, and encryption efficiency in logic encryption. Traditional logic locking pre-SAT attack has achieved structural security and encryption efficiency but not locking robustness. Post-SAT logic encryption has achieved locking robustness and encryption efficiency but not structural security.

In this work, we have investigated thoroughly the design space in logic encryption, including locking, obfuscation, and efficiency. We first established a contention between query complexity and error number in logic locking, and designed efficient locking methods that achieve both high query complexity and error number.

We then defined and solved the logic obfuscation problem. We have shown that a logic obfuscation based on the concept of universal circuit has solved the indistinguishability Logic Obfuscation (iLO). In order to achieve encryption efficiency, thus to resolve the trilemma in logic encryption, we have proposed solutions based on quasi-universal circuits such as convolutional circuits, and solutions based on the biased target circuit.

The future work should be focused on the efficient logic encryption for those target circuits with balanced on- and off-set.

Acknowledgement

This work is partially supported by NSF under CNS-1441695, CCF-1533656, and CNS-1651695.

References

[1] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 20:1–20:16, 2007.

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

[3] D. Apon, Y. Huang, J. Katz, and A. J. Malozemoff. Implementing cryptographic program obfuscation. In *CRYPTO*, 2014.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[5] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design and Test*, 27(1), 2010.

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

[7] J. Cook, O. Etesami, R. Miller, and L. Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In O. Reingold, editor, *TCC*, LNCS 5444, pages 521–538, 2009.

[8] S. A. Cook and H. J. Hoover. A depth-universal circuit. *SIAM J. Comput.*, 14:833–839, 1985.

[9] O. Coudert. On solving covering problems. In *DAC*, 1996.

[10] S. Dupuis, P.-S. Ba, G. D. Natale, M.-L. Flottes, and B. Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *IEEE International On-Line Testing Symposium*, pages 49–54, 2014.

[11] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proc. IEEE Conference on Decision and Control*, pages 40–49, Oct 2013.

[12] O. Goldreich. Candidate one-way functions based on expander graphs. *ECCC*, 7(90), 2000.

[13] R. Karmakar, S. Chatopadhyay, and R. Kapur. Encrypt flip-flop: A novel logic encryption technique for sequential circuits. *arXiv preprint arXiv:1801.04961*, 2018.

[14] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Logic encryption: A fault analysis perspective. In *DATE*, pages 953–958, 2012.

[15] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Security analysis of logic obfuscation. In *DAC*, pages 83–89, 2012.

[16] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. Fault analysis-based logic encryption. *IEEE Transactions on Computers*, 64(2), 2015.

[17] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou. Cyclic locking and memristor-based obfuscation against CycSAT and inside foundry attacks. In *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, pages 85–90, 2018.

[18] J. A. Roy, F. Koushanfar, and I. L. Markov. EPIC: Ending piracy of integrated circuits. In *DATE*, 2008.

[19] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu. Atpg-based cost-effective, secure logic locking. In *2018 IEEE 36th VLSI Test Symposium (VTS)*, pages 1–6. IEEE, 2018.

[20] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Pan, and Y. Jin. Cyclic obfuscation for creating SAT-unresolvable circuits. In *GLSVLSI*, Banff, AB, Canad, May 2017.

[21] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin. AppSAT: Approximately deobfuscating integrated circuits. In *HOST Symposium*, pages 95–100, 2017.

[22] Y. Shen and H. Zhou. Double dip: Re-evaluating security of logic encryption algorithms. In *GLSVLSI*, pages 179–184, 2017.

[23] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *HOST Symposium*, pages 137–143, 2015.

[24] L. G. Valiant. Universal circuits (preliminary report). In *ACM STOC*, pages 196–203, 1976.

[25] J. B. Wendt and M. Potkonjak. Hardware obfuscation using puf-based logic. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, pages 270–277. IEEE Press, 2014.

[26] Y. Xie and A. Srivastava. Mitigating SAT attack on logic locking. In *CHES*, pages 127–146, 2016.

[27] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu. Hardware security and trust: Logic locking as a design-for-trust solution. In *The IoT Physical Layer*, pages 353–373. Springer, 2019.

[28] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu. SARLock: SAT attack resistant logic locking. In *HOST Symposium*, pages 236–241, 2016.

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

[30] M. Yasin, A. Sengupta, M. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu. Provably-secure logic locking: From theory to practice. In *CCS*, 2017.

[31] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran. What to lock?: Functional and parametric locking. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 351–356. ACM, 2017.

[32] M. Yasin and O. Sinanoglu. Evolution of logic locking. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2017.