Attack on a Scheme for Obfuscating and Outsourcing SAT Computations to the Cloud

Khazam Alhamdan, Tassos Dimitriou and Imtiaz Ahmad
Department of Computer Engineering, Kuwait University, Kuwait

Keywords: SAT, Outsourcing Computations, Obfuscation, Data Privacy, Attack, Cloud Computing.

Abstract: The emergence of cloud computing gave users the capability to offload computations that cannot be executed locally to cloud servers with large computational power. One such computationally demanding problem is solving large satisfiability (SAT) instances. Although many problems from AI, circuit verification, etc. can be converted to SAT, outsourcing SAT instances may leak considerable information that can put a user's security at risk. Hence the privacy of outsourcing computations to the cloud is a major issue. In this work we look at the techniques of Qin et al. (Qin and Jia., 2014; Qin and Du., 2018) which have been used to obfuscate SAT formulas before they are released to the cloud. We came up with a realistic attack against their technique that demonstrates how a malicious cloud provider can obtain significant information about the underlying SAT instance. Our work shows that ad hoc schemes cannot offer the required security guarantees for outsourcing SAT computations, hence more formal frameworks should be used instead.

1 INTRODUCTION

Cloud computing is a new paradigm that enables service providers to share resources (storage, computation, bandwidth, etc.) with users. Cloud computing comes with mainly three different flavors: SaaS (Software as a service), PaaS (Platform as a service), and IaaS (Infrastructure as a service) (Mell and Grance, 2011). While each model has its own usage, outsourcing computations mainly falls under the SaaS category in which the provider offers the applications that can be run on the cloud, and the clients access these applications through an appropriate interface without having to worry about maintaining the actual hardware system. However, despite the increased convenience of cloud services, users may be reluctant to offload critical tasks to the cloud as this may leak private information to the cloud provider. It is necessary therefore to protect user's data and tasks against a malicious or untrusted provider (Zissis and Lekkas., 2012).

There are already many problems that benefit from outsourcing various computational tasks to the cloud such as those involving matrix multiplication (Atallah and Frikken, 2010) and Linear Programming (C. Wang and Wang, 2011). Other known computationally demanding problems are those related to modular exponentiation (Ding and Choo., 2017; Zhu and Chen., 2018), graph theoretic problems (Blanton et al., 2013), rule-based machine learning (Wong et al., 2007), and so on. For a recent survey on security issues in computation outsourcing, please refer to (Shan et al., 2018).

Problems such as AI planning, deduction, and software checking are also known to be hard especially when the inputs are really large. Researchers use different algorithmic techniques to solve such problems, one of which it to reduce the problem from its original form into a form (eg. propositional logic) which can be tested for satisfiablity (SAT), i.e. find an assignment to the variables that makes the formula true (Hung and Gao., 2014). Once the problem is converted to a SAT instance, various optimized and well studied SAT solvers can be used to obtain the desired solution (Rintanen, 2011; Hung and Gao., 2014).

Despite the advantage of reducing a difficult problem to such a well studied problem as SAT, still finding a solution that satisfies the propositional logic problem is not easy. This problem is also known as she satisfiability problem (SAT). Knowing that SAT is hard means it typically requires a large amount of computational power, which can be easily provided through the use of cloud services. Although outsourcing computations to the cloud solves the problem of clients with limited computational resources to some extend, security issues arise since the input-output

Alhamdan, K., Dimitriou, T. and Ahmad, I.
Attack on a Scheme for Obfuscating and Outsourcing SAT Computations to the Cloud.
DOI: 10.5220/0007829503560362
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 356-362
ISBN: 978-989-758-378-0

relationship of SAT instances may reveal considerable information about the user's activities as captured by the SAT formula. For example, SAT is often used as a tool to accomplish other goals; Wakrime (Wakrime, 2017) developed a system that considers sharing data in a privacy preserving manner using SAT. Brun et al. (Brun and Medvidovic., 2012) used SAT as an abstract view of how computation privacy can be achieved on the cloud.

Our scope here is the privacy of the problem's data, in particular of SAT formulas. One of the many approaches to provide such service is by encrypting/disguising the original problem through a procedure called obfuscation. However, a study performed in (Hosseinzadeh and Leppänen., 2018) states that obfuscation increases the difficulty of reverse-engineering the problem but it does not provide any guarantees for security. Here, we will exploit such potential weaknesses in the SAT obfuscation approach used in (Qin and Jia., 2014) and (Qin and Du., 2018).

More precisely, we developed an attack against the Qin et al. SAT obfuscation schemes and found that a malicious provider may learn considerable information about the underlying SAT formula even if this is presented in an obfuscated form. The lesson to be learned is that if an obfuscation scheme is not designed carefully then solutions obtained might reveal enough information which can be used to de-obfuscate and nullify the scheme.

The remaining of the paper is organized as follows. In Section 2, related work about privacy of SAT instances is presented. In Section 3, the (Qin and Jia., 2014) approach is described and our methodology to attack the scheme is discussed. In Section 4, we validate our findings with further experimental evidence. Finally, Section 5 concludes this work.

2 RELATED WORK

There are two aspects of the SAT privacy problem, one focusing on hiding the circuit structure of a hardware design and the other focusing on the privacy of solving computationally hard SAT instances.

Keshavarz et al. (Keshavarz and Holcomb., 2018) surveyed the major techniques used to achieve intellectual property (IP) protection at the hardware level as well as their potential weaknesses. These include reverse engineering a circuit through imaging, using a working chip, or analyzing the circuit at the logical level. A clever way of preventing the imaging attack is by disguising the gates in a way that they all look the same under imaging (Rajendran and Karri., 2013). Adding more circuitry to the desired design is an additional countermeasure that can be used to prevent simple reverse engineering using a working chip.

Logic locking is a technique attempting to lock the logical functionality of the circuit unless a certain key input is provided along with the input parameters (Chakraborty and Bhunia., 2009) of the working prototype. Roshanisefat et al. (Roshanisefat and Sasan., 2018) further introduced SRCLock (SAT-Resistant Cyclic logic Locking) to overcome some weaknesses in the original practice of logic locking. However Chen et al. (Chen, 2018) countered SRCLock by enhancing prior attacks against it. Yet, Xie et al. (Xie and Srivastava., 2018) introduced a more complex way for circuit locking called Anti-SAT, which however has also been countered by the Bit-flipping attack in (Shen and Zhou., 2018).

The previous attacks demonstrate that is not easy to hide the functionality of a circuit. However, the focus of this paper is to handle the computational privacy aspect of the SAT problem which is about hiding the original SAT formula structure from the cloud solver.

Brun et al. (Brun and Medvidovic., 2012) proposed to distribute the computation of a SAT formula on multiple machines such that every machine is assigned to evaluate a different part of the formula. The sub-evaluation is then passed to the neighbouring machines. Although this method might work in theory, nothing prevents the servers from colluding in order to recover the hidden formula. Hence this method assumes a trustworthy cloud provider.

Qin et al. in (Qin and Jia., 2014) and subsequently in (Qin and Du., 2018) proposed two similar methods that attempt to obfuscate a SAT formula by embedding a secret key in them. In the next section we describe their scheme as well as our attack against it.

3 ATTACKING THE QIN ET AL. SCHEMES

We start with a brief discussion about the underlying threat model and some background information on the SAT problem. Then we describe how the target obfuscation scheme works (Qin and Jia., 2014; Qin and Du., 2018), and how our attack takes advantage of flaws in the scheme's design.

3.1 Preliminaries

Threat Model: There are two main participants in the model of outsourcing computations to the cloud, the cloud service provider and the user who wants to outsource his/her SAT instance to the cloud. The goal of

any SAT obfuscation algorithm should be to prevent the cloud solver from getting any information about the original instance from the obfuscated one. As the cloud solver is assumed to have the computational capability to solve the obfuscated formula, any solution to this should not reveal anything about the original instance.

SAT: The Satisfiability problem (SAT) asks whether there exists an assignment $x \in \{0,1\}^n$ that makes a given Boolean function f of n variables equal to 1 (True). Typically, a cloud solver expects f to be in Conjunctive Normal Form (CNF), in which case f is a conjunction of clauses, where a clause is a disjunction of literals (e.g. a variable or its negation). In what follows, n will denote the number of variables and m the number of clauses of f.

Tseitin Encoding: Tseitin Encoding (Tseitin, 1983; Eriksson and Höglund., 2014) is a way to transform any Boolean formula or circuit into its equivalent CNF form. It has been used to encode logical circuits so then they can be analyzed using different SAT techniques. The idea is that for any circuit C with inputs $\{x_1, ..., x_n\}$ and f as the output of C, let the circuit formula of C be true *iff* (if and only if "⇔") f is true. Then substitute *iff* by its equivalent Boolean expression using $\cdot$ (and), + (or), and $\bar{}$(not).

$$C \Leftrightarrow f$$
$$(\bar{C} + f) \cdot (C + \bar{f}) \tag{1}$$

This procedure can be applied recursively on any Boolean sub-formula or gate of the circuit to transform it into its equivalent CNF form.

3.2 Qin et al. Algorithm

Qin et al. (Qin and Jia., 2014; Qin and Du., 2018) described their obfuscating algorithm as mixing and combining two SAT instances (one encoding the original formula and the other a secret known only to the user) in a way that the solution of the obfuscated SAT formula can only be reversed by the user who knows the embedded secret information, and thus the solution for the original formula can be retrieved as well.

To obfuscate a formula F, the user first picks two prime numbers p and q as a key to generate a new SAT formula (which is called the *husk* formula). This formula is obtained by asserting the output of a multiplication circuit which is equal to the product of the two prime numbers p and q. Afterwards, this formula is converted to CNF form using the Tseitin encoding procedure. The obfuscation algorithm will take as input both the husk and the original formula and feed the variables of husk into clauses of the original formula in order to mix the two instances together and wipe out any signature of the variables in the original clauses. Thus, in the end, no one should be able to determine what are the variables belonging to the original formula.

In particular, if F is the original formula, H is the husk formula and G is the obfuscated formula, then G is equal to:

$$G = \tilde{F} \wedge H, \tag{2}$$

where $\tilde{F}$ is equal F after its clauses are *mixed* with H variables. A map that matches the original variables with the obfuscated formula's variables is also generated and kept with the client as secret information that can be used to recover the solution of F from G, if such solution is found by the cloud solver. We will express the solution (satisfying assignment) for G as:

$$S_G = mix\{S_F | S_H\}, \tag{3}$$

where S_G is the solution of G, S_F is a solution of F, S_H is a solution for H, and $mix\{\ \}$ is the mix effect of the obfuscation process. S_H can be further described as a Tseitin Encoding of a multiplication circuit with the primes p and q as input and $p \cdot q$ as output.

$$S_H = Tseitin\{Mult\{p,q\}\} = p|q|Mult|p \cdot q$$
$$or$$
$$S_H = Tseitin\{Mult\{q,p\}\} = q|p|Mult|p \cdot q$$

Thus, S_G can be described as shown below:

$$S_G = mix\{S_F | p|q|Mult|p \cdot q\} \tag{4}$$

As a toy example consider a "circuit" consisting of only one AND gate with two inputs a, b, and one output c. Following the Tseitin encoding, F is equal to:

$$c \iff a \wedge b \quad or$$
$$(\bar{c} + a)(\bar{c} + b)(c + \bar{a} + \bar{b})$$

Assume further that the husk formula H is equal to xy, which makes the assignment $x = y = 1$ be the only only that satisfies H. Then $\tilde{F}$ can be expressed as:

$$\tilde{F} = (\bar{c} + a)(\bar{c} + b)(\bar{c} + x)(c + \bar{a} + \bar{b} + \bar{x}),$$

where $x = 1$ was chosen randomly out of the variables of H to be injected into F and turn it into an AND gate of three inputs a, b, and x. Now the obfuscated formula becomes

$$G = \tilde{F} \wedge H.$$

Based on this, the client will use the assignment $\{x = 1, y = 1\}$ to de-obfuscate G and recover the original

formula F.

$$\begin{aligned} G_{(x=1,y=1)} &= \tilde{F} \wedge H_{(x=1,y=1)} \\ &= (\bar{c}+a)(\bar{c}+b)(\bar{c}+1)(c+\bar{a}+\bar{b}+\bar{1}) \\ &= (\bar{c}+a)(\bar{c}+b)(c+\bar{a}+\bar{b}) \\ &= F \end{aligned}$$

Hence any satisfying assignment returned by the cloud solver for G will translate into a satisfying assignment for F, once the values for the husk formula have been plugged in G.

3.3 Attack Analysis

In this section several weaknesses will be exploited and a detailed attack will be described. The intuition of the attack is that solutions found by the cloud provider may reveal enough information about the variables that are part of the original formula. Hence the mixing effect will be nullified and a malicious cloud solver can recover the original formula (or parts of it) despite the secret information embedded by the user. Below we list two operations that will be helpful to mount the attack.

Swap Bits: Define a swap operation that takes an assignment string S of the variables and swaps the values of two bits i and j.

$$\begin{aligned} S &=< s_1,..,s_i,..,s_j,..,s_n > \\ Swap(S,i,j) \rightarrow S' &=< s_1,..,s_j,..,s_i,..,s_n > \end{aligned} \quad (5)$$

Flip Bits: Define a flip operation that takes an assignment string S of n variables and negates a specified bit of S.

$$\begin{aligned} S &=< s_1,...,s_i,...,s_n > \\ Flip(S,i) \rightarrow S' &=< s_1,...,\bar{s}_i,...,s_n > \end{aligned} \quad (6)$$

and for short: $Flip(S,i) = S^i$.

In general if multiple bits would be flipped for a set of indices $X \subset \{1,2,\ldots,n\}$, we write S^X to denote the assignment generated by flipping every bit $i \in X$.

Solution Dependency and Structure. By the structural properties of the obfuscated formula S_G, any solution S_0 found by the cloud solver has a complementary solution $\bar{S}_0$ which is defined as:

$$\begin{aligned} S_0 &= mix\{s_0|p|q|Mult|p\cdot q\} \\ \bar{S}_0 &= mix\{s_0|q|p|Mult|p\cdot q\} \end{aligned}$$

This dependency is a vulnerable point an adversary can exploit to reverse engineer the whole obfuscation process. In particular, if one can determine which bits of the solution S_0 belong to s_F and which belong to S_H, then the whole obfuscation is compromised.

De-Obfuscation Attack on Multiple Solutions. In this section we explain how the adversary can de-obfuscate G by extracting information from multiple solutions found. The underlying assumption is that the original formula has more than one satisfying assignments. Since the obfuscation algorithm of Qin et al. should work independently of the number of satisfying assignments of F, we will demonstrate how this can be used to recover information about F.

Algorithms 1 and 2 below may further be used by the cloud provider to find more assignments by flipping or swapping bits of the initial solution S_0. However, the cloud solver may rely on its own means (and its computational capacity) to find more assignments, if they exist.

Algorithm 1 (Flip Search): For each of the n bits belonging to S_0 flip that bit and test if the resulting assignment is satisfying. Collect all satisfying assignments generated this way.

Using Algorithm 1, a list of solutions L will be returned where each one is a new satisfying assignment for the obfuscated formula G. Moreover, each solution is one flip away from the initial solution S_0.

Lemma 1. *If a satisfying assignment is generated from another solution S by flipping a bit i, then i must belong to* s_F.

Proof. A bit i either belongs to s_F or s_H. However, since s_H has only two satisfying assignments which are related to each other by switching the roles of primes p and q, the flipped bit cannot be part of p or q. Since the flipped bit gave rise to a new satisfying assignment, then i must belong to s_F. So, basically flipped bits always point to variables of the original formula. □

Algorithm 2 (Swap Search): For every pair of bits belonging to S_0, swap those bits and test if the resulting assignment is satisfying. Collect all satisfying assignments generated this way.

Using Algorithm 2, a list of solutions L will be returned such that each solution is two bits away from the initial one. However, as in Lemma 1, with high probability the bits cannot be part of the prime numbers. Hence, solutions generated by swapping bits, again reveal variables belonging to s_F.

We are now ready to proceed with the actual attack. The problem tackled here can be defined formally as

Given: a CNF-SAT $G = mix\{s_0|p|q|Mult|p\cdot q\}$ and three solutions S_1, S_2, and S_3.
Objective: Find which bits belong to S_F or S_H.

Because of the dependencies between the solutions, one can deduce information by XORing different satisfying assignments. Define a relation $R_{i,j}$ as being the result of taking pointwise XOR of S_i and S_j as follows:

$$R_{1,2} = S_1 \oplus S_2 = mix\{s_{F1} \oplus s_{F2} | p_1 \oplus p_2 | q_1 \oplus q_2 | \ldots\}$$
$$R_{1,3} = S_1 \oplus S_3 = mix\{s_{F1} \oplus s_{F3} | p_1 \oplus p_3 | q_1 \oplus q_3 | \ldots\}$$
$$R_{2,3} = S_2 \oplus S_3 = mix\{s_{F2} \oplus s_{F3} | p_2 \oplus p_3 | q_2 \oplus q_3 | \ldots\}$$

Lemma 2. *If some S_i & S_j have $p_i = p_j$ & $q_i = q_j$, then all the bits which are assigned true belong to the solution S_F.*

Proof. If S_i & S_j have $p_i = p_j$ & $q_i = q_j$, then S_i & S_j will differ only on s_i & s_j. Thus, $S_i \oplus S_j = mix\{s_i \oplus s_j | 00\ldots0\}$ and the only source of 1's comes from $s_i \oplus s_j$ which is part of s_F. □

Corollary 3. *There exits at least one $R_{i,j}$ such that the number of 1's (trues) is statistically high. That is it has more 1's than the other two.*

Proof. Straightforward. □

For all $R_{i,j}$'s, all the 1-bits belong to s_F with high confidence. The information gained depends on the structure of the original solution space of the original formula F. If the solutions are clustered, then their differences in term of bits is very low. However, if the solutions are scattered over the solution space, then the bits difference between solutions will be large which will increase the information gained.

4 EXPERIMENTAL ANALYSIS

The obfuscation scheme described by (Qin and Jia., 2014) was implemented following their construction that an obfuscation formula G of F and H is equivalent to:

$$G = \tilde{F} \wedge H.$$

Variable remapping was not implemented as our analysis does not make use of variable ordering or mapping. A number of random 3-SAT formulas were generated where every variable was picked with a uniform probability. A husk formula was then generated using Wallace multiplier (Purdom and Sabry,) providing two primes p and q (notice here that the size of the primes does not really matter as the attack is agnostic to the prime numbers used). The formulas generated were based on the following categorization:

Roughly nine to ten random formulas were generated from each category shown in Table 1, for a total of 55 formulas. The ratio C/V is a very well known threshold such that when it is near 4.27, the complexity of several *complete* algorithms for checking satisfiability reaches a steep peak (Kirkpatrick and Selman, 1994). So, from an empirical point of view, the "hard" instances of SAT are to be found near that threshold. Thus, decreasing the threshold's value gives rise to more satisfying assignments.

Table 1: Types of formulas generated.

Category	# Vars V	# Clauses C	Ratio C/V
1	250	1000	4.0
2	500	1850	3.7
3	750	2550	3.4
4	2000	6800	3.4
5	3000	10200	3.4
6	4000	13600	3.4

The main experiments conducted relate to flipping (Algorithm 1) and XORing solutions found (Lemma 2). The swap search (Algorithm 2) was not used as it did not generate many new solutions. In the flip search experiment, the number of additional satisfying assignments was computed by flipping variables of an initial assignment S_0 generated by a SAT solver (Niklas and Niklas, 2017). In the XORing experiment, three solutions were combined to recover bits of S_F. The results are meant to show how easy it is to find additional assignments (in order for Lemma 2 to be applied). However, as noted before, the cloud solver may use further techniques to come up with more assignments, provided they exist.

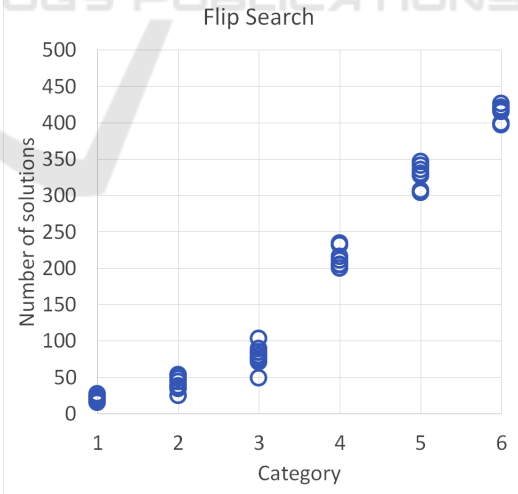

Figure 1: Flip Search Data.

Figure 1 demonstrates the number of solutions found using the Flip Search algorithm. Category 1 formulas have the least number of additional solutions found because the C/V ratio is closer to the satisfiability threshold 4.27, while category 6 formulas have the most. Hence the more solutions are found, the more

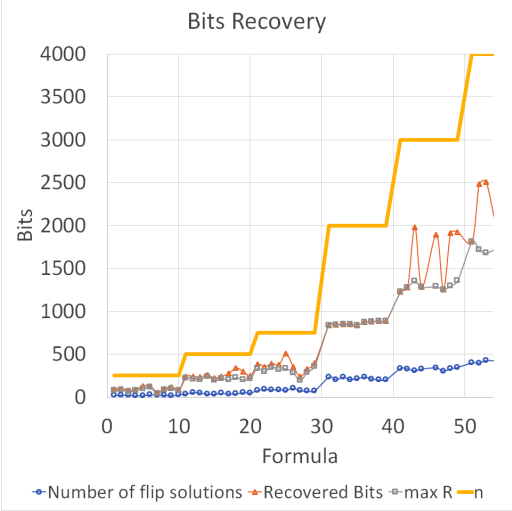

Figure 2: Total number of bits recovered by Flip Search and XORing process. The solid line denotes the number of bits (variables) in the original formula for each obfuscated one.

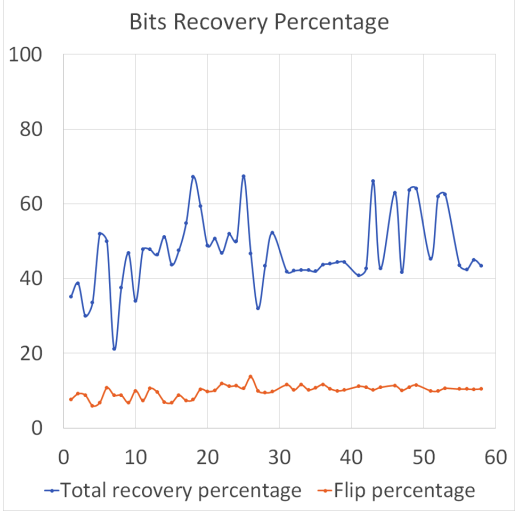

Figure 3: The percentage of bits recovered from the original formula using the XOR experiment on three satisfying assignments generated using a SAT solver.

bits are discovered to be part of the original formula.

Figure 2 demonstrates the number of bits recovered by different methods for each of the generated formulas. 'Max R' denotes the relation in which the number of 1's is the most and has an underlying solution with the same order of primes ($p_i = p_j$ & $q_i = q_j$). 'Recovered Bits', as the name suggests, denotes the combined contribution of bits found from both the XOR process and flip search.

Thus, using the analysis in Lemmas 1 and 2, many bits can be recognized to be part of the hidden formula S_F. Notice it is also possible to run the same experiments (flip-search and XOR) on the discovered solutions themselves to derive *even more* satisfying assignments.

Finally, Figure 3 demonstrates the percentage of bits found using flip search and the combined bits recovered by XORing solutions from both the SAT solver and the flip search. Although different categories of SAT formulas were used, the number of bits discovered seems to fluctuate around 50% although in some cases it is a lot more than this. Of course bit recovery depends on the similarity of solutions found, so we expect formulas with clauses-to-variables ratio closer to 4.27 to have less bits recovered. However, the vulnerability of the framework has been demonstrated as it must provide secure obfuscation for *all* SAT formulas not only those with a few satisfying assignments.

5 CONCLUSIONS

Outsourcing computations to the cloud is a new paradigm necessitated by the inherent complexity of many real life problems. One such problem is Satisfiability due to its wide applicability in many diverse applications.

In this paper we reviewed the obfuscation technique of Qin et al. (Qin and Jia., 2014; Qin and Du., 2018) for outsourcing SAT computations to the cloud and presented simple exploits that can be used by a malicious cloud provider to recover information and de-obfuscate the original formula or parts of it.

The lesson to be learned is that ad hoc techniques that try to randomize a given formula based on secret information embedded in the formula might not work well in practice. Instead, more formal frameworks should be used as for example the recent developments in Fully Homomorphic Encryption (Gentry, 2009) which allows one to securely outsource computations, or the techniques in Indistinguishability Obfuscation (Garg and Waters, 2016) where the workings of a circuit (or formula) can be provably disguised so that no reverse-engineering is possible. While these approaches have high overhead, an interesting research direction would be to see if any practical schemes to outsource SAT computations can be derived from these secure frameworks.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments and suggestions that helped improve the paper.

REFERENCES

Atallah, M. J. and Frikken, K. B. (2010). Securely outsourcing linear algebra computations. In *In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10, pages 48–59, New York, NY, USA, 2010. ACM*.

Blanton, M., Steele, A., and Alisagari, M. (2013). Data-oblivious graph algorithms for secure computation and outsourcing. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 207–218. ACM.

Brun, Y. and Medvidovic., N. (2012). Keeping data private while computing in the cloud. In *In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pp. 285-294. IEEE*.

C. Wang, K. R. and Wang, J. (2011). Secure and practical outsourcing of linear programming in cloud computing. In *In INFOCOM 2011, pages 820–828. IEEE*.

Chakraborty, R. S. and Bhunia., S. (2009). Harpoon: an obfuscation-based soc design methodology for hardware protection. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28, no. 10 (2009): 1493-1502*.

Chen, Y.-C. (2018). Enhancements to sat attack: Speedup and breaking cyclic logic encryption. In *ACM Transactions on Design Automation of Electronic Systems (TODAES) 23, no. 4 (2018): 52*.

Ding, Yong, Z. X. J. Y. and Choo., K.-K. R. (2017). Secure outsourcing of modular exponentiations under single untrusted programme model. In *Journal of Computer and System Sciences 90 (2017): 1-13*.

Eriksson, J. and Höglund., J. (2014). A comparison of reductions from fact to cnf-sat.

Garg, Sanjam, C. G. S. H. M. R. A. S. and Waters, B. (2016). Candidate indistinguishability obfuscation and functional encryption for all circuits. In *In SIAM Journal on Computing 45, no. 3 (2016): 882-929*.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *In Stoc, vol. 9, no. 2009, pp. 169-178.*

Hosseinzadeh, Shohreh, S. R. S. L. J.-M. M. J. H. S. H. and Leppänen., V. (2018). Diversification and obfuscation techniques for software security: a systematic literature review. In *Information and Software Technology*.

Hung, William NN, X. S. J. T.-X. L. J. Z. R. W. and Gao., P. (2014). Motion planning with satisfiability modulo theories. In *In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 113-118. IEEE*.

Keshavarz, Shahrzad, C. Y. S. G. X. X. and Holcomb., D. (2018). Survey on applications of formal methods in reverse engineering and intellectual property protection. In *Journal of Hardware and Systems Security 2, no. 3 (2018): 214-224*.

Kirkpatrick, S. and Selman, B. (1994). Critical behavior in the satisfiability of random boolean expressions. *Science*, 264(5163):1297–1301.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

Niklas, E. and Niklas, S. (2004-2017). Sat4j.

Purdom, P. and Sabry, A. Cnf generator for factoring problems. In *https://www.cs.indiana.edu/cgi-pub/sabry/cnf.html*.

Qin, Ying, S. S. and Jia., Y. (2014). Structure-aware cnf obfuscation for privacy-preserving sat solving. In *In Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on, pp. 84-93. IEEE*.

Qin, Ying, X. Y. S. and Du., Z. Y. (2018). Privacy-preserving sat solving based on projection-equivalence cnf obfuscation. In *In International Symposium on Cyberspace Safety and Security, pp. 224-239. Springer*.

Rajendran, Jeyavijayan, M. S. O. S. and Karri., R. (2013). Security analysis of integrated circuit camouflaging. In *In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 709-720. ACM*.

Rintanen, J. (2011). Planning with specialized sat solvers. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Roshanisefat, Shervin, H. M. K. and Sasan., A. (2018). Sr-clock: Sat-resistant cyclic logic locking for protecting the hardware. In *In Proceedings of the 2018 on Great Lakes Symposium on VLSI, pp. 153-158. ACM*.

Shan, Z., Ren, K., Blanton, M., and Wang, C. (2018). Practical secure computation outsourcing: a survey. *ACM Computing Surveys (CSUR)*, 51(2):31.

Shen, Yuanqi, A. R. and Zhou., H. (2018). Sat-based bit-flipping attack on logic encryptions. In *In Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018, pp. 629-632. IEEE*.

Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In *In Automation of reasoning, pp. 466-483. Springer, Berlin, Heidelberg*.

Wakrime, A. A. (2017). Satisfiability-based privacy-aware cloud computing. In *The Computer Journal 60, no. 12 (2017): 1760-1769*.

Wong, W. K., Cheung, D. W., Hung, E., Kao, B., and Mamoulis, N. (2007). Security in outsourcing of association rule mining. In *Proceedings of the 33rd international conference on Very large data bases*, pages 111–122. VLDB Endowment.

Xie, Y. and Srivastava., A. (2018). Anti-sat: Mitigating sat attack on logic locking. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2018).*

Zhu, Yiming, A. F. S. Y. Y. Y. S. L. and Chen., Z. (2018). New algorithm for secure outsourcing of modular exponentiation with optimal checkability based on single untrusted server. In *In 2018 IEEE International Conference on Communications (ICC), pp. 1-6. IEEE*.

Zissis, D. and Lekkas., D. (2012). Addressing cloud computing security issues. In *Future Generation computer systems 28, no. 3 (2012): 583-592*.