Estimating the Circuit Deobfuscating Runtime based on Graph Deep Learning

Zhiqian Chen
Virginia Tech
Virginia
czq@vt.edu

Gaurav Kolhe
George Mason University
Virginia
gkolhe@masonlive.gmu.edu

Setareh Rafatirad
George Mason University
Virginia
srafatir@gmu.edu

Chang-Tien Lu
Virginia Tech
Virginia
ctlu@vt.edu

Sai Manoj P.D.
George Mason University
Virginia
spudukot@gmu.edu

Houman Homayoun
George Mason University
Virginia
hhomayou@gmu.edu

Liang Zhao
George Mason University
Virginia
lzhao9@gmu.edu

ABSTRACT

Circuit obfuscation is a recently proposed defense mechanism to protect digital integrated circuits (ICs) from reverse engineering by using camouflaged gates i.e., logic gates whose functionality cannot be precisely determined by the attacker. There have been effective schemes such as satisfiability-checking (SAT)-based attacks that can potentially decrypt obfuscated circuits, called deobfuscation. Deobfuscation runtime could have a large span ranging from few milliseconds to thousands of years or more, depending on the number and layouts of the ICs and camouflaged gates. And hence accurately pre-estimating the deobfuscation runtime is highly crucial for the defenders to maximize it and optimize their defense.

However, estimating the deobfuscation runtime is a challenging task due to 1) the complexity and heterogeneity of graph-structured circuit, 2) the unknown and sophisticated mechanisms of the attackers for deobfuscation. To address the above mentioned challenges, this work proposes the first machine-learning framework that predicts the deobfuscation runtime based on graph deep learning techniques. Specifically, we design a new model, ICNet with new input and convolution layers to characterize and extract graph frequencies from ICs, which are then integrated by heterogeneous deep fully-connected layers to obtain final output. ICNet is an end-to-end framework which can automatically extract the determinant features for deobfuscation runtime. Extensive experiments demonstrate its effectiveness and efficiency.

1 INTRODUCTION

The considerable high capital costs on semiconductor manufacturing motivate most semiconductor companies to outsource their designed integrated circuits (ICs) to the contract foundries for fabrication. Despite the reduced cost and other benefits, this trend has led to ever-increasing security risks such as IC counterfeiting, piracy and unauthorized overproduction by the contract foundries [11, 15, 19, 20]. The overall financial risk caused by such counterfeit and unauthorized ICs was estimated to be over $169 billion per year [12]. The major threats from the attackers arise from reverse engineering an IC by fully identifying its functionality by stripping it layer-by-layer and extracting the unveiling gate-level netlist. To prevent such reverse engineering, IC *obfuscation* techniques have been extensively researched in recent years [1, 26]. The general idea is to camouflage some gates in an IC so that their gate types cannot be determined by reverse engineering optically, yet they preserve the functionality same as the original gates. Such techniques were highly effective until very recent progress of the attacking techniques based on logical attackers were invented and widely applied [5]. This is based on the fact that there are limited types of gates (e.g., AND, OR, XOR) in IC, so the attackers can just brute force all the possible combinations of types for all camouflaged gates to find out the one that functions identically to the targeted IC to be deobfuscated. As brute force is usually prohibitively expensive, more recently, efficient methods such as Boolean satisfiability problem (SAT)-based attacks have been proposed which have attracted enormous attention [10, 14].

The runtime of SAT attack to reverse engineer the IC highly depends on the complexity of the camouflaged IC, which can vary from milliseconds to thousands of years or more depending on the number and layout of camouflaged gates. Therefore, a successful obfuscation defense is to consume attackers prohibitive amount of time (i.e., many years) to deobfuscate. However, camouflaging each gate comes at a heavy cost in finance, power, and space, such trade-off forces us to search for optimal layout instead of purely increasing their quantity. This means to select the best set of gates to be selected for being camouflaged in order to maximize the runtime for deobfuscating. Although such selection can significantly influence the deobfuscation runtime, however, until now it is still generally based on human heuristics or experience, which is seriously arbitrary and suboptimal [7]. This is majorly because it is unable to “try and error” all the different ways of obfuscation, as there are millions of combinations to try and the runtime for each try (i.e., to run the attacker) can be days, weeks, or years.

To address this issue, this paper focuses on efficient and scalable ways to estimate the runtime for an attacker to deobfuscate a camouflaged IC. This research topic is highly under-explored

because of its significant challenges: **1) Difficulty in characterizing the hidden and sophisticated algorithmic mechanism of attackers.** Over the recent years, a large number of deobfuscation methods have been proposed with various techniques [7]. In order to practically beat the defender, methods with more and more sophisticated theories, rules, and heuristics have been proposed and adopted. The behavior of such highly-nonlinear and strongly-coupling systems is prohibitive for conventional simple models (e.g., linear regression and support vector machine [2]) to characterize. **2) Difficulty in extracting determinant features from discrete and graph-structured IC.** The inputs of the runtime estimation problem is the IC and the selected gates for camouflaging, where the first input is a heterogeneous graph while the second is a vector with discrete values. Conventional feature extraction methods are not intuitive to be applied to such type of data without significant information loss. Hence, it is highly challenging to intactly formulate and seamlessly integrate them as mathematical forms that can be input to conventional computational and machine learning models. **3) Requirement on high efficiency and scalability for deobfuscation runtime estimation.** The key to the defense against deobfuscation is the speed. The faster the defender can estimate the deobfuscation runtime for each candidate set of camouflaged gates, the more candidate sets the defender can estimate, and hence the better the obfuscation effect will be. Moreover, the estimation speed of deobfuscation runtime must not be sensitive to different obfuscation strategies in order to make the defender strategy controllable.

This work address all the above challenges, and proposes the first generic framework for deobfuscation runtime prediction, based on graph deep learning techniques. In recent years, deep learning methods in complex cognitive tasks such as object recognition and machine translation have achieved immense success, which motivates the generalization of it into graph-structured data [8]. By concretely formulating ICs and the camouflaged gates as multi-attributed graphs, this work innovatively leverages and extends the state-of-the-art graph deep learning methods such as Graph Convolutional Neural Networks (GCN) [8] to instantiate a graph regressor. Such end-to-end deep graph regressor can characterize the underlying and sophisticated cognitive process of the attacker for deobfuscating the ICs. It can also automatically extract the discriminative features that are determinants to the estimation of the deobfuscation runtime to achieve accurate runtime prediction. After being trained, the prediction based on this deobfuscation runtime estimator just runs instantly fast by simply performing a feed-forward propagation process. The major contributions of this paper are:

- Proposing a new framework, ICNet, for deobfuscation runtime estimation based on graph deep learning.
- Developing a new multi-attributed graph convolutional neural network for graph regression.
- Conducting systematical experimental evaluations and analyses on real-world datasets.

We evaluate this proof-of-concept on ISCA-85 benchmark for one replacement policy and SAT solver [21] that employs lingeling solver. However, this can be applied to any of the circuits as well as replacement policies, as the GCN learns the patterns and is not confined to any circuit or replacement policy or SAT solver.

The rest of the paper is organized as follows. Section 2 reviews existing work in this area. Section 3 elaborates a graph deep learning model for SAT runtime prediction task. In Section 4, experiments on real-world data. This paper concludes by summarizing the study's important findings in Section 5.

2 BACKGROUND AND RELATED WORK

Here, we discuss the logic obfuscation and SAT attacks followed by graph convolutional networks and the relevant works.

2.1 Logic Obfuscation and SAT Attacks

Logic obfuscation often referred as logic locking [25] is a hardware security solution that facilitates to hide the IP using key-programmable logic gates The activation of the obfuscated IP is accomplished in a trusted regime before releasing the product into the market, thereby reducing the probability to obtain the secret configuration keys by the attacker. During the activation phase, the correct key is applied to these key-programmable gates to recover the correct functionality of the IC/IP. Besides, the correct key will be stored in the IC in a tamper-proof memory.

Although obfuscation schemes try to minimize the probability of determining the correct key by an attacker, and avoid making pirated and illegal copies, introducing SAT attack shows that these schemes can be broken [21]. In order to perform SAT attack, the attacker is required to have access to the functional IC along with the obfuscated netlist. The SAT attack first tries to find the Distinguishing Input Patterns (DIP) X_i, which when applied as the input can produce different outputs (Y_i) such that ($Y_1 \neq Y_2$) when different key values are applied (K_1, K_2). This DIP can then be used to distinguish the correct and incorrect keys. The number of DIPs discovered during the SAT-based attack is the same as the number of iterations needed to unlock the obfuscated design. In each iteration, the constraint is added to SAT solver, until SAT solver cannot find a satisfying assignment. This results in finding the correct key. The SAT-based attack is summarized in the Algorithm 1.

Algorithm 1: SAT-based Attack Algorithm

1: **function** SAT_Attack(Circuit C_{obf}, Circuit C_{org})
2: $i \leftarrow 0$;
3: $F_1 \leftarrow C(X, K_1, Y_1) \wedge C(X, K_2, Y_2)$; **while** SAT$(F_i \wedge (Y_1 \neq Y_2))$ **do**
4: $X_d[i] \leftarrow$ sat_assignment $(F_i \wedge (Y_1 \neq Y_2))$;
5: $Y_d[i] \leftarrow$ eval$(X_d[i])$;
6: $F_{i+1} \leftarrow F_i \wedge C(X_d[i], K_1, Y_d[i]) \wedge C(X_d[i], K_2, Y_d[i])$;
7: $i \leftarrow i+1$;
8: $Correct\_Key \leftarrow$ sat_assignment$(F_i \mid K_1)$;
9: **end function**

Different SAT-hard schemes such as [23, 24] are proposed Furthermore, new obfuscation schemes that focus on non-Boolean behavior of circuits [22], that are not convertible to an SAT circuit is proposed for SAT resilience. Some of such defenses include adding cycles into the design [13]. By adding cycles into the design may cause that the SAT attack gets stuck in the infinite loop, however advanced SAT-based attacks such as cycSAT [28] can extract the correct key despite employing such defenses.

To ensure that the proposed defense ensures robustness against SAT attacks, the defenders need to run the rigorous simulations which could range as a step to alleviate the need to run the attack to verify whether the defense is strong enough or not. The work in [16] utilizes neural network with single-bit supervision to predict whether a given circuit in Conjunctive Normal Form (CNF) can be decrypted or not. However, this is limited to determining for few kinds of SAT-solvers, but cannot be applied to SAT-hard solutions such as SMT-SAT [27], a superset of SAT attacks. However, with proposed GCN based predictor, the defender can determine the deobfuscation time in a single run of GCN, which consumes few seconds.

2.2 Graph Convolutional Networks

Spectral graph theory is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph. Many graphs and geometric convolution methods have been proposed recently. The spectral convolution methods [4, 8] are the mainstream algorithms developed as the graph convolution methods. Their theory is based on the graph Fourier analysis [17]. The polynomial approximation is firstly proposed by [6]. Inspired by this, graph convolutional neural networks (GCNNs) ([4]) is a successful attempt at generalizing the powerful convolutional neural networks (CNNs) in dealing with Euclidean data to modeling graph-structured data. Kipf and Welling proposed a simplified type of GCNNs[8], called graph convolutional networks (GCNs). The GCN model naturally integrates the connectivity patterns and feature attributes of graph-structured data and outperforms many state-of-the-art methods significantly. With rational function, GCN can model non-smooth signal in spectral domain[3].

3 PROPOSED GRAPH LEARNING BASED SAT RUNTIME PREDICTION

This section introduces the problem setting, and present the proposed deobfuscation time prediction.

3.1 Problem Setting

First, circuit is modeled as a graph network: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is a set of n vertexes, $\mathcal{E}$ represents edges and $\mathcal{W} = [w_{ij}] \in \{0,1\}^{n\times n}$ is an unweighted adjacency matrix. A signal $x : \mathcal{V} \to \mathbb{R}$ defined on the nodes may be regarded as a vector $x \in \mathbb{R}^n$. Combinatorial graph Laplacian is defined as $\mathbf{L} = D - \mathcal{W} \in \mathbb{R}^{n\times n}$ where D is degree matrix.

Accordingly, we formulate the estimation of running time on IC as a regression task. Specifically, the model accepts graph structure along with gate attributes as input, and predict the running time:

$$Y = f(\mathcal{G}, x)\Theta, \tag{1}$$

where f is a function of graph structure $\mathcal{G}$ and x that denotes the attributes of gates such as gate type. Function f can accept heterogeneous data format for $\mathcal{G}$ and x, since $\mathcal{G}$ is often represented using matrix, while x is using vector. Θ indicates the parameters of normal neural network layers connecting the output of f and the labeled time Y, such as fully-connected layers. The goal is to learn a set of parameters of both f and Θ so that the difference between Y and $f(\mathcal{G}, x)\Theta$ is minimized.

3.2 ICNet

ICNet is a neural network that is based on graph convolution operator. As shown in Figure 1, ICNet encodes the obfuscated circuit on the left hand into two components:

- **graph structure** $\mathcal{G}$: Complete set of local connection is often used to represent the graph structure. Typically, a graph Laplacian is employed, since it contains gate-wise connection.
- **gate attributes** x : gate-level information can be encoded as numerical vector as input feature. Such information could include gate type, whether it is obfuscated and so on.

By applying the GCN, we can easily build a model to automatically learn the relationship between the circuit and deobfuscation time. However, the original graph convolutional operator is not suitable for the circuit, since the graph Laplacian will make the graph convolutional operator behavior as label propagation, i.e., the attributes of each gate are similar to its neighbors. This is called the smoothness assumption, and it does not fit the fact that gate type or encryption location of each gate does not determine its neighbors' related attributes in theory. This issue is because of graph Laplacian, which counts each node as N_i (i is the index of the row in graph Laplacian), and counts the weights of its neighbors as N_i. Consequently, they are canceled out when gate representation are aggregated using sum, and the model can hardly learn the relationship between their sum(residues) and labeled time. To solve this issue, our model employs several policies to enhance the traditional GCN for circuit learning.

- **Graph Representation** $\mathcal{G} = A$: our model uses adjacency matrix A instead of graph Laplacian. This representation can avoid intrinsic smoothness assumption which is not compatible with ICs.
- **Feature Aggregation(Θ_{feat})**: mean function is a typical methods for aggregating node feature into single number. However, mean does not consider the quantity of summed. A more flexible way is build to neural network to automatically learn feature aggregation.
- **Gate Aggregation(Θ_{gate})**: similarly, mean can also be used to aggregate gate representation into circuit graph representation. Due to the complicated real world aggregation, another neural network is designed to learn the gate aggregation function.

Our model is based on GCN setting[8] which simplify the layer parameters of graph convolutional operator and applies an approximate technique to boost the efficiency. Graph convolutional networks(GCNs), as state of the art deep learning method for the graph, focus processing graph signals defined on undirected graphs

As $\mathbf{L}$ is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors and their associated ordered real nonnegative eigenvalues identified as the frequencies of the graph. The Laplacian is diagonalized by the Fourier basis $\mathbf{U}^\intercal$: $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\intercal$ where Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e., $\Lambda_{ii} = \lambda_i$. The graph Fourier transform of a signal $x \in \mathbb{R}^n$ is defined as $\hat{x} = \mathbf{U}^\intercal x \in \mathbb{R}^n$ and its inverse as $x = \mathbf{U}\hat{x}$[17, 18]. To enable the formulation of fundamental operations such as filtering in the vertex domain, the convolution operator on graph is defined in the Fourier domain

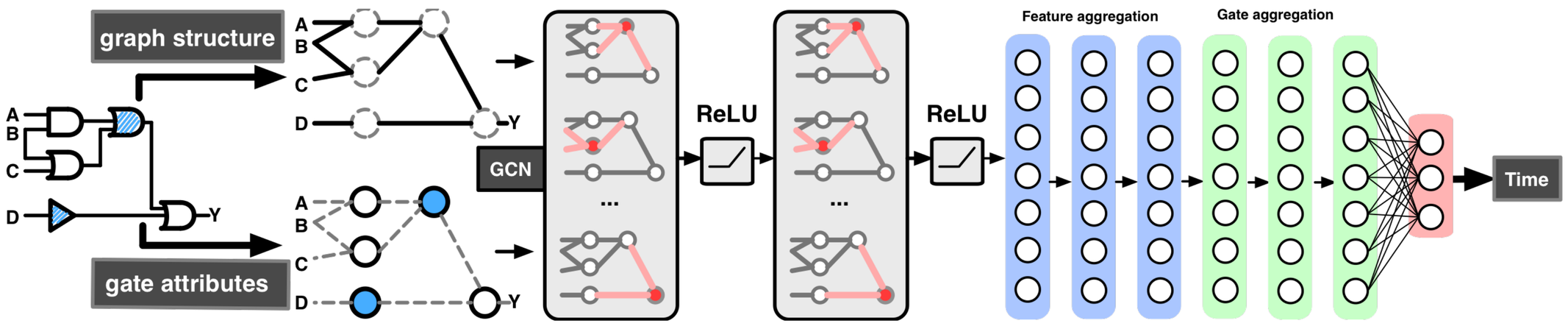


Figure 1: ICNet structure

such that $f_1 * f_2 = \mathbf{U}\,[(\mathbf{U}^\intercal f_1) \odot (\mathbf{U}^\intercal f_2)]$, where $\odot$ is the element-wise product, and f_1/f_2 are two signals defined on vertex domain. It follows that a vertex signal $f_2 = x$ is filtered by spectral signal $\hat{f}_1 = \mathbf{U}^\intercal f_1 = \mathbf{g}$ as:

$$\mathbf{g} * x = \mathbf{U}\,[\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\intercal f_2)] = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal x.$$

Note that a real symmetric matrix $\mathbf{L}$ can be decomposed as $\mathbf{L} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^{-1} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal$ since $\mathbf{U}^{-1} = \mathbf{U}^\intercal$. D. K. Hammond et al. and Defferrard et al.[4, 6] apply polynomial approximation on the spectral filter $\mathbf{g}$ so that:

$$\begin{aligned} &\mathbf{g} * x = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal x \\ &\approx \mathbf{U} \sum_k \theta_k T_k(\tilde{\mathbf{\Lambda}})\,\mathbf{U}^\intercal x \quad (\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I_N}) \\ &= \sum_k \theta_k T_k(\tilde{\mathbf{L}}) x \qquad (\mathbf{U}\,\mathbf{\Lambda}^k\,\mathbf{U}^\intercal = (\mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal)^k) \end{aligned} \tag{2}$$

According to the analysis above, graph Laplacian matrix is replaced with adjacency matrix. To fit whole-graph level regression task, the proposed method designs two aggregation neural networks. Formally, it is denoted as:

$$\begin{aligned} Y &= \mathbf{g} * x \\ &= GCN(\mathcal{W}, x)\Theta_{feat}\Theta_{gate} \\ &\approx \mathcal{W}x\Theta_{GCN}\Theta_{feat}\Theta_{gate} \qquad \text{(simplification \& approximation)} \end{aligned} \tag{3}$$

However, the running time tends to grow at an exponential rate as the number of encrypted gates increases. Therefore, the model is modified as:

$$Y = e^{\mathcal{W}x\Theta_{GCN}\Theta_{feat}\Theta_{gate}} \tag{4}$$

As shown in Fig. 1, the model conducts one or two graph convolutional operation(GCN) to fuse information from graph structure and gate attributes in the spectral domain. Then two sets of neural networks are performed for the feature and gate aggregation. Last few layers are fully connected to predict the runtime.

3.3 Algorithm description

The Algorithm 1 first prepare graph adjacency as circuit connection representation(line 2). To fit machine learning schema, the whole dataset is split into training and testing dataset. Each dataset is then split into small batch size to improve learning efficiency(line 3-4). ICNet training is an iterative process which updates the model until the residues are small enough or converged(line 6-13). First,

Algorithm 2: ICNet

Input: a integrated circuit graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, gate attribute set: $x_j(i)$, i $\in 1, 2, ..., |\mathcal{V}|$ for each encryption instance D_j, the real runtime Y_j for instance D_j

Output: a neural network function with parameters Θ_{GCN}, Θ_{feat} and Θ_{gate}

1 // data preparing
2 calculate $\mathcal{W}$ which is the adjacency matrix of $\mathcal{G}$
3 split encryption instances D into training set D_{train} and testing set D_{test}
4 split both D_{train} and testing set D_{test} into batch set d_{train} and testing set d_{test}
5 // update ICNet
6 $\theta = \{\Theta_{GCN}, \Theta_{feat}, \Theta_{gate}\}$
7 initialize θ with Gaussian or uniform distribution.
8 **repeat**
9 randomly select one $d_{train} = x_{d1}, x_{d2}, ...$
10 calculate runtime $e^{\mathcal{W}d_{train}\Theta_{GCN}\Theta_{feat}\Theta_{gate}}$ ▷ Eq. 4
11 calculate residues $\delta = Y - e^{\mathcal{W}d_{train}\Theta_{GCN}\Theta_{feat}\Theta_{gate}}$
12 compute derivatives to update parameters: $\theta \leftarrow \theta + \beta\nabla_\theta\delta$, where β is learning rate
13 **until** δ *convergence*;

the model parameters are initialized by Gaussian or uniform distribution. In each iteration, a batch of the training set is selected randomly. By equation 4, the model computes the predicted runtime(line 10) and then calculates the residues between real runtime and prediction(line 11). Following normal deep learning schema, the model update parameters by the derivatives regarding the parameters themselves with learning rate(line 12).

4 EVALUATION

This section elaborates evaluation of the proposed method ICNet with competitive baselines including: Graph deep learning methods:

- **GCN[8]**
- **ChebNet[4]**

The input of these models above is exactly same as our model. We also compare against several stat of art regression models[1]:

- Linear Regression(LR)
- LASSO

[1]https://scikit-learn.org/stable/modules/linear_model.html

- Epsilon-Support Vector Regression(SVR). Two kernels were applied: polynomial(P) and RBF(R).
- Ridge Regression(RR)
- Elastic Net(EN)
- Orthogonal Matching Pursuit(OMP)
- SGD Regression
- Least Angle Regression(LARS)
- Theil-Sen Estimators(Theil)

These regression models does not model graph using Laplacian or adjacency matrix, since they can only accept feature vector. Therefore, the input are encoded as mean or sum on concatenation of Laplacian or adjacency matrix and gate attributes.

4.1 Data processing

The datasets are obtained by running SAT algorithm [20, 21] on real-world ISCA-85 benchmark: First, we take a circuit and select a random gate and replace it with LUT of fixed size (LUT size 4 in current work). To deobfuscate, we implement SAT attack algorithm [20, 21] with the obfuscated circuit netlist as input. We monitor the time that sat takes to decode the key, which is the deobfuscation time. The proposed model is evaluated on two datasets:

- **Dataset 1**: the total number of the encryption location ranges from 1 to 350, this is for testing if the model is sensitive to the number of encrypted quantity of gates.
- **Dataset 2**: the total number of the encryption location ranges from 1 to 3, this is for testing if the model can handle very small value.

The circuit in the experiments is the same, and the total gate number of the circuit is 1529. For graph deep learning methods, graph is represented using Laplacian matrix or adjacency matrix, while for general regression baselines, the graph Laplacian or adjacency matrix is summed or averaged across gates. Though the evaluations shown here are mere proof-of-concept of how powerful the proposed GCN based deobfuscation runtime prediction is, it can be applied to a SAT-hardening solution utilizing any replacement policy, LUT size and other SAT parameters, by retraining GCN.

4.2 Experiment configuration

The features of gate used in experiments include

- **gate mask**: if the gate is encrypted, the value is set to 1, otherwise 0.
- **gate type**: the gate type include {AND, NOR, NOT, NAND, OR, XOR}, they are encoded using one-hot coding, such as [1,0,0,0,0,0,] for AND and [0,1,0,0,0,0,] for NOR gate.

For graph deep learning model(ChebNet and ICNet), the graph structure is represented using graph Laplacian matrix or adjacency matrix. These model employ ADAM [9] optimizer and will stop learning when the learning loss is converged. The implementation of our model will be available online. All the baselines and the proposed model are tested on two different feature set, since gate type is useful or not is unknown.

- **Location**: Only the gate mask is included.
- **All features**: Besides gate mask, gate type is also included.

For node aggregation, we apply *sum* and *mean* since they are the popular. Deep learning model can have another node aggregation method, i.e., learning by a neural network automatically. Therefore, in the results, $ChebNet-NN$ and $ICNet-NN$ denote the automatic version. It is expected that deep neural network can learn a the optimal aggregation which is not worse than our assumption, i.e., sum or mean.

	Location		All feat	
Method	Sum	Mean	Sum	Mean
SVR RBF	1.6791	0.6784	1.6675	0.6739
SVR Poly	0.1913	2.1890	0.1696	2.2091
SGD	2.1450e+25	2.1823	1.0430e+26	2.2072
LR	0.2839	0.2284	0.2449	0.2253
RR	0.2309	2.1508	0.2058	2.1738
LASSO	0.9213	2.1843	1.0127	2.2083
EN	0.5763	2.1843	0.6409	2.2083
OMP	1.8182	1.9192	1.8651	2.0337
LARS	1.9968	2.1277	2.0434	2.1833
Theil	0.2948	0.2238	0.2385	0.2277
ChebNet	0.1484	8.8370e+33	0.1761	0.1760
ChebNet-NN	0.17858		3.8549e+27	
GCN	0.3364	0.4149	0.2496	0.3290
GCN-NN	0.1811		0.1606	
ICNet	0.1534	0.1256	0.2390	0.1902
ICNet-NN	**0.0843**		**0.1367**	

Table 1: Regression Performance (Mean Square Error) on Dataset 1

	Location		All feat	
Method	Sum	Mean	Sum	Mean
SVR RBF	0.0051	0.0048	0.0050	0.0051
SVR Poly	0.0048	0.0048	0.0048	0.0051
SGD	7.6301e+25	0.0045	2.0675e+26	0.0049
LR	6.9063ee+23	4.6521e+20	7.2916e+25	5.8600e+23
RR	0.0070	0.0045	0.0065	0.0049
LASSO	0.0047	0.0045	0.0046	0.0049
EN	0.0047	0.0045	0.0046	0.0049
OMP	0.0047	0.0045	0.0045	0.0049
PAR	0.0054	0.1918	0.0051	0.3143
LARS	0.0047	0.0045	0.0046	0.0049
Theil	N/A	N/A	N/A	N/A
ChebNet	0.0047	0.0045	4.3570e+28	0.0048
ChebNet-NN	**0.0043**		0.0047	
GCN	0.0061	0.0046	0.0048	0.0050
GCN-NN	0.0050		0.1606	
ICNet	0.0049	0.0047	**0.0040**	0.0043
ICNet-NN	0.0051		0.0048	

Table 2: Regression Performance (Mean Square Error) on Dataset 2

4.3 Regression Results

In the dataset 1 experiment(Table 1, all methods achieved acceptable mean square error except SGD (sum) which did not learn a reasonable model to predict the runtime, since the value is very large (at e+25/+26 scale). Most regression methods are sensitive to the aggregation method. For example, only using location feature, MSE of RR is 0.2319 when using sum, but it got 2.1508 when using

mean function. Sensitive models include SVR, LASSO, and EN. The best of the regression baselines is LR and Theil, which achieved around MSE of 0.22. On the other hand, graph deep learning model ChebNet is slightly better than the best regression model. However, ChebNet is not stable and sensitive to the aggregation method and feature set, since it may yield a very large error. Our model, ICNet, is stable to the feature and aggregation setting and outperformed all the other methods, i,e, 0.11001 of MSE. Note that ICNet-NN is better than ICNet with the sum or mean, which demonstrates that there exists a better aggregation method, and deep neural network can learn this function automatically. Note that ICNet is always better than GCN under any settings, which shows that our improvement works on circuit senario.

While in the dataset 2, it is more challenging, since all the runtime is small and the model has to be very precise to achieve low MSE. All methods at almost the same level of MSE. Once again, some of the regression models are not stable such as SGD and LR. Graph deep learning method includes ChebNet and ICNet still at the best error level. ChebNet can achieve the best level but sensitive to the settings, while ICNet is insensitive to all configuration. ICNet-NN is still the best method, and it outperformed its mean and sum version. Moreover, ICNet is more stable than GCN and ChebNet.

4.4 Prediction behavior analysis

In the section, we show several predicted value along with real value to analyze the prediction characterization.

Since there is little difference in dataset 2, we choose several competitive baselines in dataset 1 experiments. Several baselines performed very badly such as OMP and SGD which only output values around a constant level. SVR(RBF) is also bad and yield constant value when the real runtime is larger than a threshold. The results of EN and LASSO is positively related to the real values, but the correlation parameters are significantly different from the truth. Linear, RR, SVR(POLY) and Theil predicted the values that are relatively closer than that of the other baselines, but with high variance. The proposed method, ICNet, not only predicted the value very precisely but also with small variance.

5 CONCLUSION

In this work, we have introduced a neural network model for recovering SAT runtime on ICs. To properly fuse graph structure and gate attributes, an enhanced graph convolutional operator is introduced. The proposed method can avoid attribute propagation which is in the original GCN but not suitable for ICs. Experiments on real-world datasets suggest that the proposed model is capable of modelling the runtime regarding the circuit graph accurately.

REFERENCES

[1] Kimia Zamiri Azar, Hadi Mardani Kamali, Houman Homayoun, and Avesta Sasan. 2019. SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), 97–122.

[2] Christopher M Bishop and Tom M Mitchell. 2014. Pattern Recognition and Machine Learning. (2014).

[3] Zhiqian Chen, Feng Chen, Rongjie Lai, Xuchao Zhang, and Chang-Tien Lu. 2018. Rational Neural Networks for Approximating Jump Discontinuities of Graph Convolution Operator. *arXiv preprint arXiv:1808.10073* (2018).

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

[5] Mohamed El Massad, Siddharth Garg, and Mahesh V Tripunitara. 2015. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes.. In *NDSS*.

[6] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[7] Soroush Khaleghi and Wenjing Rao. 2018. Hardware Obfuscation Using Strong PUFs. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 321–326.

[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[9] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. 2011. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 265–272.

[10] Duo Liu, Cunxi Yu, Xiangyu Zhang, and Daniel Holcomb. 2016. Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*. IEEE, 433–438.

[11] P. D. Sai Manoj, Ferdinand Brasser, L. Davi, A. Dhavlle, T. Frassetto, S. Rafatirad, A. Sadeghi, A. Sasan, H. Sayadi, S. Zeitouni, and H. Homayoun. 2018. Hardware-Assisted Security: Understanding Security Vulnerabilities and Emerging Attacks for Better Defenses. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*.

[12] IHS Technology Press Release: Top 5 most counterfeited parts represent a $169 billion potential challenge for global semiconductor industry. [n. d.]. https://technology.ihs.com/405654/top5-most-counterfeited-parts-represent-a-169-billion-potentialchallenge-for-global-semiconductor-market, 2. http://www.test.org/doe/

[13] Shervin Roshanisefat, Hadi Mardani Kamali, and Avesta Sasan. 2018. SRCLock: SAT-Resistant Cyclic Logic Locking for Protecting the Hardware. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI (GLSVLSI '18)*.

[14] Shervin Roshanisefat, Harshith K Thirumala, Kris Gaj, Houman Homayoun, and Avesta Sasan. 2018. Benchmarking the Capabilities and Limitations of SAT Solvers in Defeating Obfuscation Schemes. *arXiv preprint arXiv:1805.00054* (2018).

[15] H. Sayadi, N. Patel, P. D. Sai Manoj, A. Sasan, S. Rafatirad, and H. Homayoun. 2018. Ensemble Learning for Hardware-based Malware Detection: A Comprehensive Analysis and Classification. In *ACM/EDAA/IEEE Design Automation Conference*.

[16] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2018. Learning a SAT Solver from Single-Bit Supervision. *ArXiv* abs/1802.03685 (2018).

[17] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.

[18] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. 2016. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis* 40, 2 (2016), 260–291.

[19] J. Stangl, T. Loruenser, and P. D. Sai Manoj. 2018. A Fast and Resource Efficient FPGA Implementation of Secret Sharing for Storage Applications. In *ACM/EDAA/IEEE Design Automation and Test in Europe (DATE)*.

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

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

[22] Y. Xie and A. Srivastava. 2017. Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction. In *ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[23] Y. Xie and A. Srivastava. 2018. Anti-SAT: Mitigating SAT Attack on Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018), 1–1.

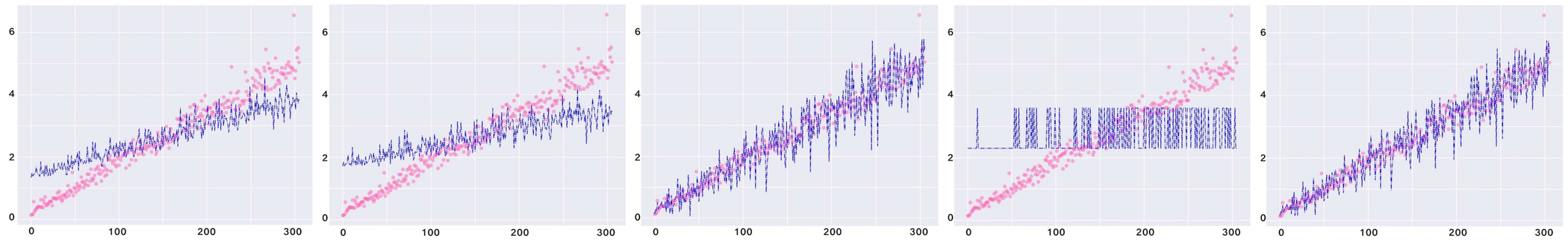
[24] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu. 2016. SARLock: SAT attack resistant logic locking. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*.

[25] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri. 2016. On Improving the Security of Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 9 (Sept 2016), 1411–1424.

[26] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan JV Rajendran, and Ozgur Sinanoglu. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1601–1618.

[27] Kimia Zamiri Azar, Hadi Mardani Kamali, Houman Homayoun, and Avesta Sasan. 2019. SMT-Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond The SAT Attacks. In *Transaction of Cryptography Hardware and Embedded Systems*.

[28] H. Zhou, R. Jiang, and S. Kong. 2017. CycSAT: SAT-based attack on cyclic logic encryptions. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 49–56.



(a) EN (b) LASSO (c) Linear (d) OMP (e) RR

(f) SGD (g) SVR(Poly) (h) SVR(RBF) (i) Theil (j) ICNet

Figure 2: Comparison between predictions and real values: Pink dot are real values, blue lines are the predictions. x-axis is data index in testing data while y-axis is runtime value