**QUEEN'S
UNIVERSITY
BELFAST**

A Novel Feature Extraction Strategy for Hardware Trojan Detection

Yu, S., Gu, C., Liu, W., & O'Neill, M. (2020). A Novel Feature Extraction Strategy for Hardware Trojan Detection. In *The IEEE International Symposium on Circuits and Systems (ISCAS): Proceedings* (IEEE International Symposium on Circuits and Systems (ISCAS): Proceedings). Institute of Electrical and Electronics Engineers Inc.. <https://doi.org/10.1109/ISCAS45731.2020.9180479>

Published in:

The IEEE International Symposium on Circuits and Systems (ISCAS): Proceedings

Document Version:

Peer reviewed version

Queen's University Belfast - Research Portal:

[Link to publication record in Queen's University Belfast Research Portal](#)

Publisher rights

Copyright 2020 IEEE. This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

General rights

Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

A Novel Feature Extraction Strategy for Hardware Trojan Detection

Shichao Yu*, Chongyan Gu*, Weiqiang Liu†, Maire O'Neill*

*Centre for Secure Information Technologies, ECIT, Queen's University Belfast, Belfast, UK

†College of EIE, Nanjing University of Aeronautics and Astronautics, Nanjing, China

E-mails: {syu08, c.gu}@qub.ac.uk, liuweiqiang@nuaa.edu.cn, m.oneill@ecit.qub.ac.uk

Abstract—Hardware Trojans (HTs) are acknowledged as a significant emerging security concern in the IC industry resulting from the globalization of the semiconductor supply chain. Recently, taking advantage of the exponential growth in computing power, machine learning (ML) approaches such as neural networks (NNs) are being considered for HT detection. However, the circuit structure and components of an IC design are different from the data types in the ML models. To efficiently extract HT features from complex IC designs and utilize common ML-based detection approaches is challenging. In this paper, a novel HT feature extraction strategy based on gate-level circuit netlists is proposed to tackle the challenges. The HT features are extracted from the circuit topology rather than statistical analysis in previous research. A commonly utilized support vector machine (SVM)-based HT detection model is employed for data training and testing using the extracted features on HT benchmarks from both open-sourced library and HT generation platform to prove the feasibility and efficiency of the proposed HT feature extraction strategy. The detection results show high recall in nearly all tested benchmarks, achieving at most 97.7% recall on sequential Trojans and 84.8% on combinational ones.

Index Terms—Hardware Trojans, feature extraction, netlist, circuit topology, directed graph, structural features, machine learning

I. INTRODUCTION

Due to the increasing complexity of Integrated Circuits (ICs), nowadays, it is more difficult for a single vendor to complete the entire manufacturing of chips. The design and fabrication of ICs is now distributed worldwide. However, the use of overseas foundries, test facilities and third party vendors increases the risk of IC products being attacked by Hardware Trojans (HTs).

HTs are malicious tampering of ICs at any untrusted phase of the production chain. They can be small malicious circuits inserted in chips or malicious modifications to the chips. Fig. 1 shows the typical structure of a functional HT [1]. The trigger circuitry monitors a set of normal signals and activates the malicious payload when trigger conditions are satisfied. As most HTs have rare trigger conditions and the size of trigger circuitry is small, common IC tests and verification workflows can fail to detect HTs [2] [3].

Meanwhile, taking advantage of the exponential growth in computing power, ML methods have shown huge potential for HT detection in recent research [1], [4], [5], [6].

A gate-level netlist is a generic IC design file that describes the network topology and layout of a circuit and is provided by

Fig. 1. The key components of a HT circuit [1].

third-party vendors as the data exchange format for intellectual property (IP).

However, the data structure and basic component of a gate-level netlist are significantly different from the commonly used data types in the ML models. It is challenging to efficiently extract HT features from netlists and using common ML algorithms for HT detection.

When comparing netlists with sequential data like audio traces and text, netlists not only have dependency in the data (instance) order, but also have multi-connections between instances. When compared with 2-dimensional image data, the positional information of netlist elements is different from the absolute position of image pixels. The position information of instances is relative and can only be retrieved from the net signals between instances. So data parsing and feature extraction in gate-level netlists is much more complex and difficult than common data types in ML.

Statistical analysis has been applied in existing research on ML-based HT detection to select netlist features [6], [7]. These features performed well to detect some HTs. However, to recognize a different type of Trojan, the circuit features had to be re-selected to gain an equivalent performance or prevent detection failure.

Furthermore, statistical analysis of netlist features does not involve a direct analysis of the circuit topology (Directed Graph), and as such, circuit structural information is lost in the process.

To address these problems, we propose a novel HT feature extraction strategy that extracts structural features from the network topology of gate-level netlists. In the experiment, a commonly utilized SVM-based HT detection model is employed for data training and testing, using the extracted relative positional structure features.

To the best of our knowledge, this is the first time that the

relative positional relationship of a netlist graph is investigated as features for ML-based HT detection.

The main contributions of this paper are as follows:

- 1) A novel feature extraction strategy based on circuit topology rather than statistical analysis is proposed.
- 2) A pin-level feature searching algorithm in netlist block is proposed for the feature extraction.
- 3) The application of the extracted pin-level feature traces on a commonly utilized SVM model for HT detection.

The remainder of this paper is organized as follows: Section II discusses the related Trojan feature extraction work. The proposed feature extraction strategy and its implementation are presented in Section III. Section IV presents the experimental results and Section V provides some concluding remarks.

II. PREVIOUS RESEARCH ON GATE-LEVEL TROJAN FEATURE EXTRACTION

For gate-level HT detection at the design-stage, [8] first adopted the controllability and observability feature of nets in gate-level netlist for HT classification using K-means clustering. [9] adopted controllability as the feature and reduced the false positive rate by signal switching probability simulation.

Circuit level statistics (i.e. the number of primitives, AND and OR gates) are utilized along with controllability and observability in [10] to make a four-dimensional vector to train a SVM-based Trojan classifier. In [5], 5 HT classification features were identified after statistical analysis of features in normal and Trojan circuit. [11] extracted the similar features of [5] from the trigger nets of HTs and improved the HT recognition accuracy under an ensemble-learning-based method.

In [6], 11 Trojan-net features were selected from 51 netlist features for HT detection using the random forest algorithm. Neural networks were first proposed for use in HT detection on gate-level netlists in [7], using features from their earlier research work [5], [6].

Recently, [12] proposed two-level AONN (AND, OR, NAND, NOR) gates as a feature for combinational HT and extracted the number of register-to-register and input-to-register paths as the feature for Deep State Machine (DSM) Trojans.

In summary, most of the previous research selects HT features by statistical analysis of netlist features, which does not consider the structural features of HTs. Although [12] adopted structural features for combinational HT detection, it is limited to some specific hardware structures.

However, in the proposed feature extraction strategy, we do not focus on specific hardware structures. All of the pin-level structural information within gate-level netlists will be extracted as features for HT detection. The details of the proposed strategy are explained in the next section.

III. THE PROPOSED FEATURE EXTRACTION STRATEGY

As ML-based HT detection methods classify Trojan circuits by features, we extract features on known HT benchmarks from both Trust-hub [13] and HT generation platform in [1].

Starting from each instance in the netlist, the proposed feature extraction strategy extracts all structural features from

Fig. 2. (a) The schematic of gate-level netlist slice from a Trojan trigger.; (b) The pin-level directed graph of path 1 in (a).

the corresponding netlist blocks in the topology network of gate-level netlist.

A. The Topology of a Gate-level Netlist

In order to extract structural features, gate-level netlists are parsed at first. Fig. 2(a) is the schematic of a gate-level netlist slice from the trigger part of a sequential FSM (finite-state machine)-based HT [1].

As shown in Fig. 2(a), each instance in the netlist is a basic logic cell (cells are defined in the cell library). The structural information of the netlist is composed of two parts: the cell type of each instance and the connection information between instances.

Using instances as the nodes and net names as the edges, the topology of a netlist can be drawn as a pseudo-graph in which both loops and multiple edges are permitted. In order to simplify the proposed searching algorithm, we separate all the pins belonging to each instance as independent nodes to generate a directed graph. Fig. 2(b) shows the pin-level graph layout of path 1 in the netlist in Fig. 2(a) with independent pin nodes. The generated pin nodes are named as *cell_pin* for both input and output pins.

The network topology of the netlist has been built as a directed graph. The cell node name refers to the cell type of each instance while the connection between pin nodes refers to the connection information between instances.

B. Pin-level Structural Features

Based on the generated topology of the netlist in subsection III-A, structure features in pin-level can be extracted by BFS (Breadth-First-Search) searching algorithm for graphs. As shown in TABLE I, the pin-level structure of Path 1 in Fig. 2(a) is extracted from the pin-level netlist graph in Fig. 2(b).

Features extracted from different netlists should be comparable. If two designs are compiled in different cell libraries (A , B), a mapping table from cell library A to cell library

TABLE I
THE PIN-LEVEL STRUCTURAL FEATURES OF PATH 1 FROM FIG. 2(B)

Name	Cell→Pin Out→Pin In→Cell	Corresp. Net
U1	(None, INPUT, NOR2_IN2, NOR)	g2
	(NOR2, NOR2_OUT, NOR_IN1, NOR2)	g1
	(NOR2, NOR2_OUT, DFF_D, DFF)	g5
U7	(NONE, INPUT, DFF_CLK, DFF)	clk
	(DFF, DFF_Q, NOR2_IN1, NOR2)	g11
U2	(NAND2, NAND2_OUT, NOR2_IN2, NOR)	g10
	(NOR2, NOR2_OUT, OUTPUT, None)	g12

Fig. 3. Program architecture of pin-level structural feature extraction for HT detection.

```

*cell: U194 → Instance Name
**Side: L → Input Side Paths Group
U204, nor2s3, Q, n309, U194, nnd2s3, DIN1, 4, → Pin-level Structure Path
U207, nnd2s3, Q, n317, U204, nor2s3, DIN1, 3,
U352, i1s3, Q, n89, U207, nnd2s3, DIN1, 2,
\ DFF_69/Q_reg, sdfs1, QN, n458, U352, i1s3, DIN, 1,
U208, nnd2s3, Q, n288, U207, nnd2s3, DIN2, 2,
U209, nor2s3, Q, n289, U208, nnd2s3, DIN1, 1,
U384, i1s3, Q, n121, U208, nnd2s3, DIN2, 1,
U406, ib1s9, Q, n139, U204, nor2s3, DIN2, 3,
input, input, input, G14, U406, ib1s9, DIN, 2,
U385, i1s3, Q, n122, U194, nnd2s3, DIN2, 4,
\ DFF_28/Q_reg, sdfs1, QN, n456, U385, i1s3, DIN, 3,
input, input, input, CK, \ DFF_28/Q_reg, sdfs1, CLK, 2,
U193, nnd2s3, Q, G321, \ DFF_28/Q_reg, sdfs1, DIN, 2,
U195, nnd2s3, Q, n307, U193, nnd2s3, DIN1, 1,
U194, nnd2s3, Q, n308, U193, nnd2s3, DIN2, 1,
\ DFF_27/Q_reg, sdfs1, Q, n496, \ DFF_28/Q_reg, sdfs1, SDIN, 2,
input, input, input, CK, \ DFF_27/Q_reg, sdfs1, CLK, 1,
U196, nnd2s3, Q, G315, \ DFF_27/Q_reg, sdfs1, DIN, 1,
\ DFF_26/Q_reg, sdfs1, Q, n495, \ DFF_27/Q_reg, sdfs1, SDIN, 1,
input, input, input, test_se, \ DFF_27/Q_reg, sdfs1, SSEL, 1,
input, input, input, test_se, \ DFF_28/Q_reg, sdfs1, SSEL, 2,
**Side: R → Output Side Paths Group
U193, nnd2s3, DIN2, n308, U194, nnd2s3, Q, 4,
\ DFF_28/Q_reg, sdfs1, DIN, G321, U193, nnd2s3, Q, 3,
\ DFF_29/Q_reg, sdfs1, SDIN, n497, \ DFF_28/Q_reg, sdfs1, Q, 2,
\ DFF_30/Q_reg, sdfs1, SDIN, n498, \ DFF_29/Q_reg, sdfs1, Q, 1,
U171, or3s3, DIN2, n441, \ DFF_29/Q_reg, sdfs1, QN, 1,
U173, or2s3, DIN2, n441, \ DFF_29/Q_reg, sdfs1, QN, 1,
U177, xnr2s3, DIN1, n441, \ DFF_29/Q_reg, sdfs1, QN, 1,
U385, i1s3, DIN, n456, \ DFF_28/Q_reg, sdfs1, QN, 2,
U194, nnd2s3, DIN2, n122, U385, i1s3, Q, 1,
U230, or5s3, DIN2, n122, U385, i1s3, Q, 1,
U232, nnd4s2, DIN1, n122, U385, i1s3, Q, 1,

```

Fig. 4. Pin-level netlist block of s1423_T426 U194.

B should be created to support HT detection on netlists from different cell libraries.

C. The Implementation of the Proposed Strategy

The program architecture for pin-level structural feature extraction is presented in Fig. 3. First, the Pin-Level Graph Generator loads the netlist and builds a pin-level directed graph for the netlist according to subsection III-A.

Then, netlist blocks around each instance are extracted from the graph based on the BFS algorithm. The searching depth is

TABLE II
PIN-LEVEL STRUCTURAL FEATURE TRACE (U194, *Logic_level* = 4)

Index 0	1	2	3
L_{-3} PCP	L_{-2} PCP	L_{-1} PCP	L_0 PCP
DIN,i1s3,Q	DIN1,nnd2s3,Q	DIN1,nor2s3,Q	DIN1,nnd2s3,Q
4	5	6	7
L_1 PCP	L_2 PCP	L_3 PCP	Label
DIN2,nnd2s3,Q	DIN,sdfs1,Q	SDIN,sdfs1,Q	Trojan/Normal

defined by the value of parameter *logic_level*. For example, if the current instance is U_n and *logic_level* is defined as 4, the module will search and extract all the pin connections and instances around U_n in 4 logic levels. The sample of an extracted netlist block from s1423_T426, instance U194 [13] is shown in Fig. 4. For each line in Fig. 4, the pin-level path is formed as:

$$\underbrace{\text{Instance, Cell, Pin, Net}}_{\text{Next Logic Level}}, \underbrace{\text{Instance, Cell, Pin, Logic_level}}_{\text{Current Logic Level}}$$

Each netlist block is filtered by the pin-level feature filter. Starting from the centre instance, all the pin-level structural paths in netlist block from and to the centre instance will be extracted. After library mapping for netlists from different libraries and data labelling based on the known HT instances, each pin-level structural feature trace for U_n can be extracted.

TABLE II shows one feature trace belonging to U194. The basic element of a pin-level feature trace is L_n PCP, where L_n means the current PCP is n logic levels to the centre instance (L_0 is the centre instance), while PCP is (*pin, cell, pin*) and contains the name of the input pin, cell and output pin from the same instance on feature trace. For example, L_{-3} PCP (*DIN, i1s3, Q*) in TABLE II means the input pin *DIN* and output pin *Q* from cell *i1s3* is on the pin-level feature trace, 3 logic levels away to the input of centre instance U194.

When all netlist blocks in a netlist are processed, each instance will have corresponding list of pin-level structural feature traces. As these extracted traces contain all the relative structure information surrounding the centre instance, they are used as the relative structural features of each instance for both Trojan instances and normal instances and can be applied to ML-based classification model for HT detection.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

The proposed HT feature extraction strategy was implemented in Python. To prove the feasibility and efficiency of the strategy and control the training and hyperparameter tuning cost, a general C-Support Vector Machine [14]-based HT detection model is designed for data training and testing using the extracted features on HT benchmarks from both Trust-hub [13] (group 1, 2, 3) and the HT generation platform in [1] (group 4), to prove the feasibility and efficiency of the proposed strategy.

To keep the relative positional relationship in the extracted feature traces, we utilize data array from index 0 to 6 in TABLE II (*Logic_level* = m , PCP $L_{-(m-1)}$ to L_{m-1}) as a 7-dimensional feature and index 7 as the label for

TABLE III
EXPERIMENTAL RESULT OF SVM-BASED HT DETECTION BY THE PROPOSED PIN-LEVEL STRUCTURAL FEATURES

Training Set/ Testing Set	LL	C	Class Weight	#of TPs	#of FNs	PRC	REC	Training Set/ Testing Set	LL	C	Class Weight	#of TPs	#of FNs	PRC	REC
Group 1 c2670_T007/ c2670_T008 Combinational Trojan	2	1.0	None	26	17	81.2	60.4	Group 3 s13207_T428/ s1423_T426 Sequential Trojan	3	1.0	None	633	76	92.6	89.2
			Balance	30	13	9.5	69.7				Balance	659	50	45.1	92.9
		1.5	None	28	15	82.3	65.1			1.5	None	639	70	89.7	90.1
			Balance	30	13	9.5	69.7				Balance	659	50	49.1	92.9
	3	1.0	None	53	24	77.9	68.8		4	1.0	None	2213	187	77.4	92.2
			Balance	64	13	55.6	83.1				Balance	2214	186	69.5	92.2
		1.5	None	55	22	78.5	71.4			1.5	None	2213	187	77.4	92.2
			Balance	62	15	61.3	80.5				Balance	2214	186	73.5	92.2
Group 2 s1423_T426/ s1423_T424 Sequential Trojan	3	1.0	None	643	253	71.4	97.7	Group 4 s13207_comb5/ s15850_comb4 Combinational Trojan	2	1.0	None	0	19	0	0
			Balance	840	56	65.9	93.7				Balance	16	3	0.8	84.2
		1.5	None	745	151	95.5	83.1			3.0	None	0	19	0	0
			Balance	836	60	70.1	93.3				Balance	16	3	1.2	84.2
	4	1.0	None	2333	705	89.5	76.7		3	1.0	None	19	14	10.9	57.5
			Balance	2432	606	75.2	80.1				Balance	28	5	1.9	84.8
		1.5	None	2344	694	89.7	77.1			3.0	None	18	15	4.8	54.5
			Balance	2428	610	77.7	79.9				Balance	28	5	2.2	84.8

training and testing. One-hot coding is used to encode pin and cell. Each PCP in feature dataset is translated into a vector, $\{pin_vector, cell_vector, pin_vector\}$. The label is encoded as “0” for a normal PCP trace, and “1” for a Trojan trace.

B. Results and Analysis

The C-SVM model is trained with a Linux virtual machine with a 4-core i7-6700 CPU and 8 GB RAM. As the balanced class weight requires more multiplications, it takes 60 ms/100-traces for training and 40 ms/100-traces for classification. The corresponding figures for the unweighted model are 7 ms/100-traces and 5 ms/100-traces.

TABLE III shows the classification results, *Training Set* is the netlist from which we extract feature traces for training, while *Testing Set* for testing. *LL* is the *Logic_level* corresponding to the searching depth and length of each feature trace. *C* is the penalty parameter of the misclassification in SVM model. As the number of Trojan class feature traces is much smaller than normal ones in each netlist, we balanced the weight of each class using a *Class Weight* option. The number of TPs (True Positives) shows the number of real Trojan traces identified to be Trojan traces. The Number of FNs (False Negatives) shows the number of Trojan traces identified as normal trace. PRC (precision) is the fraction of real Trojan traces among the traces identified to be Trojan, while REC (recall) equal the fraction of the identified Trojan traces among all Trojan traces ($REC = TP / (TP + FN)$).

From TABLE III, we find that the proposed features, where result in a high recall in nearly all groups (at most 97.7% on sequential Trojans and 84.8% on combinational ones). The balanced class weight makes the classification model more robust, but also allows more normal traces to be misclassified as Trojans. Increasing penalty parameter *C* can slightly improve both the precision and recall.

For different HT types, each one has an ideal logic-level for feature extraction to get the best precision and recall (3

for group 1, 2, 4 and 4 for group 3). And the classification results on sequential trojan groups perform much better than combinational ones. This phenomenon can be understood as the combinational Trojan has fewer structural features than the sequential Trojan in this experiment. In particular, the combinational Trojans in group 4, generated from HT generation platform [1], only have AND gates in their logic, which makes them contain less structural features than all other groups. This results in the lowest precision on classification and causing classification failure.

In conclusion, the extracted structural features can be successfully applied in ML models for HT detection. High recall results can be obtained in detecting Trojan traces in most HT benchmark netlists. Also, with more structural features from a Trojan circuit, the adopted SVM model can obtain a better classification result.

V. CONCLUSION

In this paper, we propose a new feature extraction strategy for hardware Trojan detection. We first build a pin-level directed graph for a netlist and then extract the relative pin-level structural feature from the network topology of the netlist rather than using statistical analysis as in previous research. A SVM-based HT detection model is designed to test the extracted features. The experimental results show that the extracted features can be successfully applied in ML models for HT detection and high recall can be achieved for most Trojan benchmark netlists.

In the future, we will improve the proposed feature extraction strategy and ML-model to obtain better detection results for Trojans with less structural features.

ACKNOWLEDGMENT

This research is supported by RISE (ukrise.org - EP/R011494/1) and funded by the UK Engineering and Physical Sciences Council (EPSRC) and the National Cyber Security Centre (NCSC).

REFERENCES

- [1] S. Yu, W. Liu, and M. O'Neill, "An improved automatic hardware Trojan generation platform," in *Proc. IEEE Computer Society Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 302–307.
- [2] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware Trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, vol. 1, no. 1, pp. 85–102, Mar 2017.
- [3] M. Xue, R. Bian, J. Wang, and W. Liu, "Building an accurate hardware Trojan detection technique from inaccurate simulation models and unlabelled ICs," *IET Computers Digital Techniques*, vol. 13, no. 4, pp. 348–359, 2019.
- [4] M. Xue, R. Bian, W. Liu, and J. Wang, "Defeating untrustworthy testing parties: A novel hybrid clustering ensemble based golden models-free hardware Trojan detection method," *IEEE Access*, vol. 7, pp. 5124–5140, 2019.
- [5] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *Proc. IEEE 22nd Int. Symp. On-Line Testing and Robust System Design*, July 2016, pp. 203–206.
- [6] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.
- [7] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. On-Line Testing and Robust System Design*, July 2017, pp. 227–232.
- [8] H. Salmani, "Cotd: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 338–350, Feb 2017.
- [9] Y. He and K. Huang, "Trigger identification using difference-amplified controllability and dynamic transition probability for hardware Trojan detection," *IEEE Trans. Inf. Forensics Security*, pp. 1–1, 2019.
- [10] X. Xie, Y. Sun, H. Chen, and Y. Ding, "Hardware Trojans classification based on controllability and observability in gate-level netlist," *IEICE Electronics Express*, vol. 14, no. 18, pp. 20 170 682–20 170 682, 2017.
- [11] Y. Wang, T. Han, X. Han, and P. Liu, "Ensemble-learning-based hardware Trojans detection method by detecting the trigger nets," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, May 2019, pp. 1–5.
- [12] Q. Liu, P. Zhao, and F. Chen, "A hardware Trojan detection method based on structural features of Trojan and host circuits," *IEEE Access*, vol. 7, pp. 44 632–44 644, 2019.
- [13] H. Salmani and M. Tehranipoor, "Trust-Hub," accessed on 2019-10-02. [Online]. Available: <https://trust-hub.org/home>
- [14] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.