ランダムフォレストに基づくハードウェアトロイ検知に対する強化学習を用いた自律的なハードウェアトロイ生成手法

池上 裕香[1,a)]　根岸 良太郎[1]　長谷川 健人[2]　披田野 清良[2]　福島 和英[2]　戸川 望[1]

概要：近年，IC 設計工程でハードウェアトロイ (HT) と呼ばれる不正回路が挿入されるリスクが深刻となっている．HT の検知には機械学習を用いる手法が有効であるとされるが，どのような HT の検知が可能か，あるいは検知が難しいか検討が不十分である．検知の難しい HT を効率的に生成する必要がある．本稿ではランダムフォレストに基づく HT 検知を対象に，強化学習を用いた自律的な HT 生成手法を提案する．提案手法では強化学習を用いることにより，効率的で自律的な HT 生成が可能となる．提案手法で用いる強化学習アルゴリズムは，新たな HT の生成と HT 検知精度の観測を繰り返し，HT 検知において検知精度を最小化するよう，HT 構成時のパラメータ変更の方策を学習する．ランダムフォレストに基づく HT 検知を対象とした評価実験の結果，提案手法は HT 検知精度を低下させる HT を構成することを確認した．

キーワード：ハードウェアトロイ，ゲートレベルネットリスト，強化学習

Hardware Trojan Generation against ML-based Hardware Trojan Detection

Yuka Ikegami[1,a)]　Ryotaro Negishi[1]　Kento Hasegawa[2]　Seira Hidano[2]　Kazuhide Fukushima[2]
Nozomu Togawa[1]

Abstract: In this paper, we propose a hardware-Trojan (HT) generation method utilizing reinforcement learning against random-forest-based HT detection. The proposed method learns how to change the parameterized HTs so as to minimize the HT detection rate. Experimental evaluation results confirm the effectiveness of the proposed method.

Keywords: Hardware Trojan, Gate-level netlist, Reinforcement learning

1. はじめに

近年，様々な機器の IoT (Internet of Things) 化が進んだことで IC の需要が高まり，ハードウェア製品の開発においては IC を安価で効率的に生産することの重要性が増している．安価で効率的に IC を生産するための方法として，設計や製造工程の一部がサードパーティに委託されている [1]．IC の設計・製造段階では，設計書の作成に始まり，IP コアの設計やレイアウト設計，製造などの段階が存在しており，各段階でサードパーティが関与する可能性がある．

IC の設計・製造段階でサードパーティが関与する中で，悪意のある第三者にハードウェアトロイ (HT) を挿入されるリスクが報告されている [2], [3]．HT とは，回路の内部

[1] 早稲田大学基幹理工学研究科情報理工・情報通信専攻
Dept. Computer Science and Communications Engineering, Waseda University
[2] 株式会社 KDDI 総合研究所
KDDI Research, Inc.
[a)] yuka.ikegami@togawa.cs.waseda.ac.jp

状態がある条件を満たした時に，情報漏洩や機能改変などの製造元が意図しない動作を発現する回路である．特に，回路設計工程における回路設計情報への HT の挿入が指摘されており，機械学習を用いた HT 検知手法が有効とされている．機械学習を用いた HT 検知では，機械学習モデルとして，ニューラルネットワークを用いる手法 [4] や，ランダムフォレストを用いる手法 [5]，グラフ学習を用いる手法 [6] が存在する．いずれも高い精度で HT の存在，または HT に該当する部分を検知できることが報告されている．中でも文献 [7] では，機械学習モデルとしてランダムフォレストを対象に，回路中の各ネットに対する特徴量セットを最適化することで，TPR 0.733，TNR 1.000，Precision 0.970 と非常に高い HT 検知を達成している．しかし，文献 [5] では Trust-HUB に公開されているベンチマーク回路に対する検知精度の評価にとどまっており，より検知が難しい HT に対する検知性能は保証されていない．HT 検知手法を実際に活用するためには，日々更新される検知が難しい HT に対して適応できる必要がある．

さて，強化学習とは，人間が手動で設計することが難しいような方策を，エージェントと環境の相互作用を通じて学習するものである．強化学習により自律的に HT を生成することで，日々更新される検知が難しい HT に対し，効率的に HT の検知精度を向上できる．強化学習を利用して HT を生成する手法として，文献 [8], [9] が提案されているが，ランダムフォレストを対象とした評価は行われていない．

本稿では，ランダムフォレストに基づく HT 検知手法 [7] を対象に，強化学習を用いた自律的な HT 生成手法を提案する．さらに，生成された HT から，ランダムフォレストに基づく HT 検知において検知が困難となる条件を考察する．

本稿の貢献は以下の通りである．

(1) ランダムフォレストに基づく HT 検知を対象に，強化学習を用いた自律的な HT 生成手法を提案する．強化学習により行動の方策を自律的に学習し，ランダムフォレストにより状態を効果的に観測する．
(2) ランダムフォレストに基づく HT 検知を対象に，強化学習が効果的に学習するよう状態，行動，報酬を設計する．
(3) 評価実験を通じ，提案手法により検知が困難な HT サンプルを生成することを示す．

本稿の構成を以下に示す．2 章で提案手法で用いる強化学習とランダムフォレストを用いた HT 検知を説明する．3 章でランダムフォレストに基づく HT 検知に対する，強化学習を用いた自律的な HT 生成手法を説明する．4 章で提案手法を用いて生成したサンプルに対し，HT の検知精度を評価する．5 章で本稿をまとめる．

2. 強化学習と HT 検知

本章では，提案手法で用いる強化学習とランダムフォレストを用いた HT 検知を説明する．

2.1 強化学習

強化学習は，エージェントと呼ばれる行動主体が特定の環境内で学習を通じて最適な行動を見つける手法である [10]．エージェントは環境と相互作用を行い，現在の状態を観測して次に取るべき行動を選択する．その結果，環境から報酬を受け取ることができる．強化学習の目標は，累積される報酬を最大化するための最適な方策を見つけることである．

強化学習の基本的な数学的モデルにマルコフ決定過程 (MDP)，$M = (S, A, P, R)$ がある．S はエージェントが環境内で取り得る状態の集合，A はエージェントが各状態で取ることができる行動の集合，$P : S \times A \times S \to [0, 1]$ は状態遷移確率関数，$R : S \times A \to \mathbb{R}$ は即時報酬関数を示す．時刻 t において，エージェントは環境から現在の状態 $s_t \in S$ と報酬 r_t を受け取る．次に，方策 $\pi(a|s)$ に基づいて行動 $a_t \in A$ を実行する．ここで方策 $\pi(a|s)$ とはある状態 s において行動 a を選択する確率を表す．環境は新しい状態 s_{t+1} に移動し，それに応じた報酬 r_{t+1} が決定される．強化学習では，このような環境とエージェントの相互作用を通じて，割引累積報酬 R_t の期待値を最大化する方策 π を学習する．エピソードが時刻 t から T まで続くときの割引累積報酬は，割引率 $\gamma \in (0, 1]$ を用いて，$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k}$ で表される．エピソードはエージェントが実行する一連の行動を示す．

強化学習を利用して HT を生成すれば，機械学習モデルにとって検知するのが困難な HT を生成することが期待できる．

2.2 ランダムフォレストに基づく HT 検知

HT を検知するために，機械学習を用いた手法の研究が進んでいる．中でも，1 章で議論したように特徴量セットを最適化したランダムフォレストは，Precision がほぼ 1.00 で誤りなく HT を構成するネットを検知できる点において，極めて検知精度が高い．本稿では，ランダムフォレストに基づく HT 検知手法を対象とする．

2.2.1 HT 検知

機械学習による HT 検知は，HT に含まれるネット (トロイネット) か HT に関わらない通常のネット (ノーマルネット) かを示すラベルが付加されたネットリストをもとに，ネットリストに含まれる各ネットに対してまず特徴量を抽出する．抽出された特徴量を用いて HT を検知するモデルを生成する．次にテスト対象であるネットリストの各

表 1 実験で使用した特徴量.

#	Feature	#	Feature
1	out_nearest_pout	21	fan_in_u5d4
2	fan_in_u5d1	22	fanin5
3	fan_in_u4d1	23	in_flipflop_3
4	fan_in_u3d1	24	fan_in_u4d3
5	fan_in_u5d2	25	out_multiplexer_5
6	out_nearest_multiplexer	26	out_flipflop_3
7	out_nearest_flipflop	27	fan_in_u4d5
8	fan_in_u5d3	28	in_flipflop_2
9	fan_in_u4d2	29	fan_in_u1d4
10	fan_in_u2d1	30	fan_in_u4d4
11	in_nearest_multiplexer	31	fan_in_u3d3
12	out_flipflop_5	32	fan_in_u3d2
13	out_flipflop_4	33	in_loop_4
14	in_nearest_pin	34	fan_in_u2d2
15	fan_in_u1d5	35	fan_in_u1d3
16	out_flipflop_2	36	fan_in_u1d2
17	fan_in_u1d1	37	fanin1
18	in_nearest_flipflop	38	fanin4
19	fan_in_u5d5	39	fan_in_u2d5
20	in_flipflop_4		

ネットから抽出された特徴量を，生成されたモデルに与えることで，ネットリストに含まれる各ネットがトロイネットかノーマルネットかを識別する．

2.2.2 ランダムフォレストに基づく HT 検知手法と HT 特徴量

これまで多くの HT を特徴付けるネットリスト特徴量が提案されている [5], [11]．文献 [7] では，文献 [5], [11] で提案されたすべてのネットリスト特徴量を対象にランダムフォレストにおける特徴量重要度を求め，そのうち 39 個のネットリスト特徴量が最適な特徴量セットであるとした．本稿では，文献 [7] で提案された 39 個の特徴量を使用する．使用した特徴量を表 1 に示す．

3. 提案手法

本章では，ランダムフォレストに基づく HT 検知に対する，強化学習を用いた自律的な HT 生成手法を提案する．図 1 に提案手法の概要を示す．提案手法は HT 構成部，HT 検知部，強化学習部の 3 つの処理から構成される．そのうち，HT 構成部，HT 検知部は環境が，強化学習部はエージェントが担う．これら 3 つの処理を繰り返し実行することで，検知の難しい HT を生成する．

3.1 HT 構成部

一般的に，HT はトリガ回路とペイロード回路から構成される．トリガ回路は，回路の内部状態がトリガ回路に設定された条件を満たしているかを判定する回路である．回路の内部状態が条件を満たしている場合，トリガ回路はペイロード回路を有効にする．トリガ回路によって起動されたペイロード回路は，実際に情報漏洩や性能低下などの悪意ある機能を発現する．

表 2 トリガ回路のテンプレート.

回路名	機能
組合せ	組合せ回路で構成されたトリガ回路. トリガ条件を即時的に判定する.
順序	順序回路で構成されたトリガ回路. トリガ条件を一定期間満たすか判定する.

HT 構成部では，まず強化学習部から得られるパラメータに基づいてトリガ回路とペイロード回路を構成する．生成されたトリガ回路とペイロード回路を，予め与えられている通常回路の任意の箇所に接続することで，HT のサンプルを生成する．HT 構成部における手順を以下に示す．

(1) 以下の要素を準備する．
- 通常回路 C_n
- トリガ回路のテンプレート T_t
- トリガ回路のテンプレートに設定するパラメータの集合 P_t
- ペイロード回路のテンプレート T_p
- ペイロード回路のテンプレートに設定するパラメータの集合 P_p
- トリガ入力の信号線の集合 W_t
- ペイロード入出力の信号線の集合 W_p

(2) トリガ回路の構成と接続
- (a) トリガ回路のテンプレート T_t に対して，パラメータ P_t を適用することでトリガ回路 C_t を構成する．
- (b) 構成されたトリガ回路 C_t と通常回路 C_n を，信号線 W_t で接続する．

(3) ペイロード回路の構成と接続
- (a) ペイロード回路のテンプレート T_p に対して，パラメータ P_p を適用することでペイロード回路 C_p を構成する．
- (b) 構成されたペイロード回路 C_p をトリガ信号の信号線でトリガ回路 C_t と接続し，信号線 W_p で通常回路 C_n と接続する．

(4) 通常回路 C_n，トリガ回路 C_t，ペイロード回路 C_p をまとめて 1 つの回路を構成し，サンプル回路 C を生成する．

トリガ回路とペイロード回路のテンプレートとして実装した回路と機能を，それぞれ表 2 と表 3 に示す．これらはレジスタ転送レベル (RTL) で記述する．生成されたサンプル回路 C は，論理合成によりゲートレベルのネットリストへ変換される．RTL 記述の論理合成には yosys[*1] を利用し，論理合成の標準セルライブラリには SAED 90nm CMOS ライブラリを利用した．

[*1] https://github.com/YosysHQ/yosys

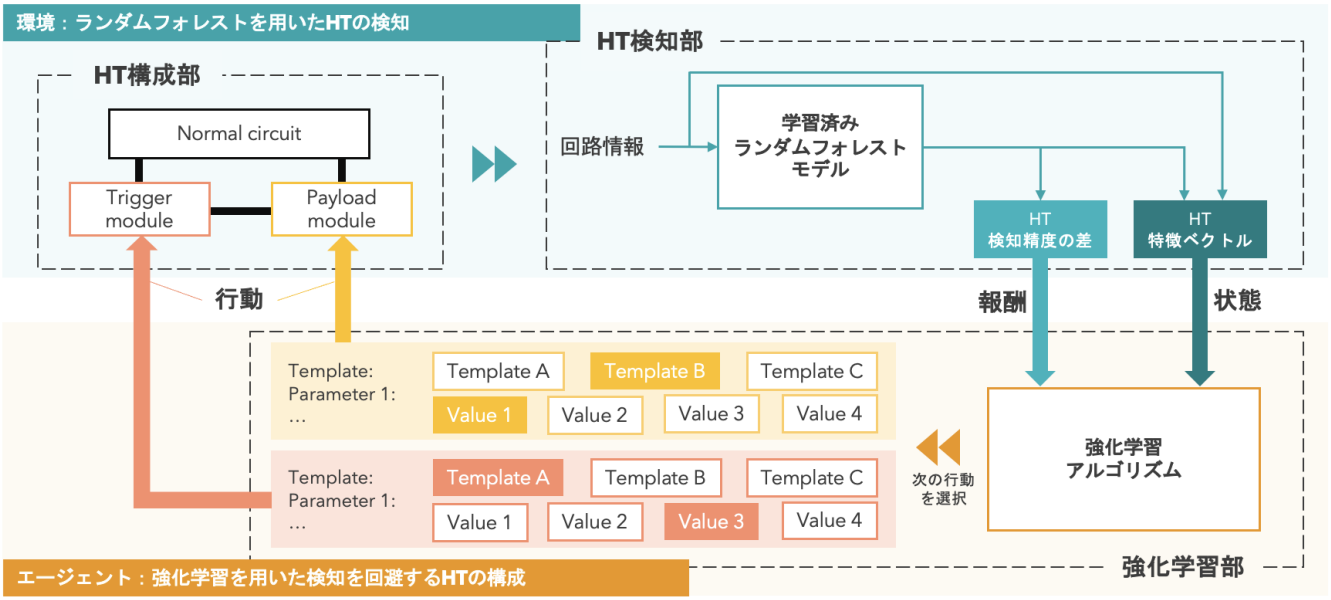

図 1 提案手法の構成.

表 3 ペイロード回路のテンプレート.

回路名	機能
電力消費	リングオシレータにより電力を消費する.
DoS	信号線の値を 0 で上書きする.
情報流出	出力される信号線の値を指定の内部信号の値で上書きする.

3.2 HT 検知部

HT 検知部では，HT 構成部で生成されたサンプルを用いて，学習済みのランダムフォレストモデルにより HT を検知する.

まず，HT 構成部で生成されたサンプルから特徴量を抽出する．検知に用いる特徴量は，サンプル回路を構成する各ネットについて，表 1 に示す 39 個の特徴量である．次に，抽出した特徴量を学習済みのランダムフォレストモデルに入力し，識別結果を出力させる.

その識別結果を用いて，HT の検知精度を算出する．HT の検知精度には，正しくトロイネットとして分類されたトロイネットの割合を示す再現率 (Recall) を用いる．正しくトロイネットとして分類されたトロイネットの数を TP (True Positive)，誤ってノーマルネットとして分類されたトロイネットの数を FN (False Negative) とすると，Recall は $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ で求められる.

HT の検知精度の算出の他に，エージェントに状態として渡す特徴ベクトルを生成する．特徴ベクトルは，ランダムフォレストモデルに入力したネットリスト特徴量と出力された識別結果を連結して生成される.

3.3 強化学習部

強化学習部では，生成したサンプルに対する HT 検知の Recall が目標値を下回るよう，HT 検知部から渡された HT 検知精度と特徴ベクトルに基づいて，次の行動となるサンプル生成時に必要なパラメータを選択する.

強化学習における学習の手順を以下に示す.

(1) 最初の HT サンプル C_{org} ならびに目標とする Recall p_{trgt} を準備する.
(2) サンプル回路 C を C_{org} で初期化する.
(3) p に 1，p' にサンプル回路 C に対する HT の検知精度を設定する.
(4) HT の検知精度 p' が目標値 p_{trgt} を下回るまで以下の手順を繰り返す.
  (a) 報酬 r に HT の検知精度の差 $\exp(p)-\exp(p')$ を設定.
  (b) 状態 s と報酬 r に基づき，方策 π を更新し，次の行動 a を算出.
  (c) 行動 a に基づき，HT 構成部で HT サンプルを構成.
  (d) p を p' で更新.
  (e) HT 検知部から受け取った C に対する特徴ベクトルと検知精度で s と p' を更新.

強化学習の結果，モデルが学習された後，学習済みモデルを用いて HT のサンプルを生成する場合は，方策を更新せずに，現在の状態に基づいて次の行動を算出する.

本稿では，状態，行動，報酬を以下のように定める.

状態 HT 検知部で得られる特徴ベクトル.
行動 HT 構成部で HT のサンプルを構成するためのパラメータの設定.
報酬 HT 検知部で得られる HT 検知精度の差.

以降，本節では，提案する強化学習における状態，行動，報酬を説明する.

表 4 HT 構成のためのエージェントの行動.

#	操作
1	トリガ回路のテンプレートを組合せ回路に設定.
2	トリガ回路のテンプレートを順序回路に設定.
3	ペイロード回路のテンプレートを電力消費回路に設定.
4	ペイロード回路のテンプレートを DoS 回路に設定.
5	ペイロード回路のテンプレートを情報流出回路に設定.
6	トリガ入力の信号線を 1 つ増やす.
7	トリガ入力の信号線を 1 つ減らす.
8	順序回路においてトリガ条件となる入力回数を 1 増やす.
9	順序回路においてトリガ条件となる入力回数を 1 減らす.
10	ペイロード入出力の信号線を 1 増やす.
11	ペイロード入出力の信号線を 1 減らす.

3.3.1 状態

状態は，HT 検知部で得られる特徴ベクトルを用いる．ランダムフォレストモデルに入力した特徴量と出力された識別結果を連結させたものを特徴ベクトルとすることで，HT 全体の特徴を表現できる．

3.3.2 行動

行動は，HT 構成部で HT のサンプルを構成するためのパラメータの設定を用いる．パラメータ設定として，トリガ回路やペイロード回路として用いるテンプレート (T_t, T_p) の選択，それぞれのテンプレートにおける内部パラメータの値 (P_t, P_p) の増減，トリガ回路やペイロード回路が通常回路と接続するための信号線の数，信号線の候補の中から実際に用いる信号線 (W_t, W_p) の選択がある．具体的な行動の一覧を表 4 に示す．

3.3.3 報酬

報酬は，HT 検知部で得られる HT 検知精度の差を用いる．時刻 t において HT 検知部で算出される Recall を p_t とすると，その時点での報酬は $p_{t-1} - p_t$ で求められる．HT 検知精度が大きく低下するほど，報酬がより大きくなる．逆に，HT 検知精度が向上すれば，負の報酬を得る．このように報酬を設定することで，検知の難しい HT を生成するよう学習させることができる．

そこで，提案手法では，p_t が十分に大きいときに精度を大きく低下させる行動を高く評価するため，$r_t = \exp(p_{t-1}) - \exp(p_t)$ で報酬を算出する．

4. 評価実験

本章では，提案手法により学習したモデルを用いてサンプルを生成し，HT の検知精度を評価する．

4.1 実験条件

本節では実験条件として，ランダムフォレストや強化学習で使用したデータセットやパラメータ，および検証方法を説明する．

4.1.1 ランダムフォレスト

表 5 にランダムフォレストの学習時に使用したネットリストと，各ネットリストのトリガ回路およびペイロード回路のテンプレートの種類，ノーマルネット数およびトロイネット数を示す．表 5 における#1–#30 は，Trust-HUB [12], [13] で公開されている RS232 回路をもとに作成した HT である．この 30 個の HT は，RS232 の通常回路に対してランダムに不正回路を挿入して生成した．ランダムフォレストモデルの訓練には#1–#20 の 20 個の HT を使用し，テストには#21–#30 の 10 個の HT を使用した．パラメータは文献 [7] で提案されたものを使用し，表 6 に示す．10 個の HT に対する平均の Recall は 0.803 であり，通常の HT 検知においては十分な精度である．

4.1.2 強化学習

強化学習アルゴリズムには PPO [14] を用い，実装には stable-baselines3[*2]を利用した．Recall の目標値は 0.6 と設定した．強化学習モデルの学習時には，2048 ステップの学習を行い方策を訓練した．

4.1.3 検証方法

強化学習により方策を学習したモデルを用いて HT を 100 個生成し，各 HT において 10 ステップ以内に Recall が目標値 0.6 を下回るか検証する．次に，生成した 100 個の HT に対する Recall の平均と，強化学習を適用していない 10 個の HT に対する Recall の平均を比較し評価する．

4.2 実験結果

実験の結果，100 個のうち 86 個において，10 ステップ以内で Recall が 0.6 未満となる HT を生成できた．提案手法により生成した 100 個の HT に対する Recall の平均と，強化学習を適用していない 10 個の HT に対する Recall の平均を表 7 に示す．表 7 より，強化学習を適用していない場合は Recall の平均が 0.803 であるのに対し，提案手法を用いた場合は Recall の平均が 0.541 となった．また，Recall の最小値は 0.444 であり，100 個のうち 8 個の HT において Recall が 0.444 となった．提案手法により検知が困難な HT サンプルを生成できることが確認できた．

学習済みモデルを用いて生成した HT から，HT 検知において検知が困難となる条件を考察する．今回生成した HT は，すべてトリガ回路は ‘組合せ’，ペイロード回路は ‘情報流出’ であった．ただし，各回路の内部パラメータの値はそれぞれ異なる．生成した HT のうち，トリガ回路とペイロード回路の構成の一例を図 2 に示す．エージェントが取る行動として，まずトリガ入力の信号線をいくつか増やした上で，トリガ回路やペイロード回路を ‘組合せ’ や ‘情報流出’ に設定する傾向にあった．トリガ入力の信号線を増やすという行動が選択されたのは，トリガ入力の信号線が増えるほど，トリガが有効になる条件が厳しくなり，HT 検知においてより検知が難しくなるからである．組合せ回

[*2] https://github.com/DLR- RM/stable- baselines3

表 5 実験に使用したネットリスト.

#	ネットリスト	トリガ回路	ペイロード回路	ノーマルネット	トロイネット
1	RS232-seed0	順序	電力消費	236	316
2	RS232-seed1	組合せ	電力消費	252	89
3	RS232-seed2	組合せ	DoS	239	6
4	RS232-seed3	組合せ	DoS	242	7
5	RS232-seed4	組合せ	電力消費	251	89
6	RS232-seed5	順序	情報流出	244	160
7	RS232-seed6	組合せ	電力消費	252	92
8	RS232-seed7	順序	情報流出	244	160
9	RS232-seed8	組合せ	電力消費	250	88
10	RS232-seed9	順序	電力消費	235	328
11	RS232-seed10	組合せ	情報流出	244	6
12	RS232-seed11	順序	電力消費	235	314
13	RS232-seed12	順序	電力消費	250	317
14	RS232-seed13	順序	情報流出	244	159
15	RS232-seed14	組合せ	電力消費	236	88
16	RS232-seed15	組合せ	電力消費	250	88
17	RS232-seed16	順序	電力消費	250	319
18	RS232-seed17	順序	電力消費	250	310
19	RS232-seed18	組合せ	DoS	255	7
20	RS232-seed19	組合せ	情報流出	241	4
21	RS232-seed100	組合せ	情報流出	240	6
22	RS232-seed101	組合せ	DoS	242	7
23	RS232-seed102	組合せ	情報流出	241	3
24	RS232-seed103	順序	電力消費	251	320
25	RS232-seed104	組合せ	電力消費	250	87
26	RS232-seed105	順序	DoS	252	157
27	RS232-seed106	順序	情報流出	243	162
28	RS232-seed107	組合せ	DoS	252	3
29	RS232-seed108	組合せ	DoS	242	4
30	RS232-seed109	順序	DoS	245	170

表 6 ランダムフォレストモデルの構築時に使用したパラメータ.

パラメータ	値
n_estimators	644
max_depth	29
max_features	9
min_samples_leaf	1
min_samples_split	6

表 7 実験結果.

対象の HT	Recall
提案手法により生成した 100 個の HT	0.541
強化学習を適用していない 10 個の HT	0.803

路はその時の入力のみで出力が決定するため設計が比較的単純であるのに対し，順序回路はその時の入力と以前の状態で出力が決定するため複雑な状態遷移やタイミングの問題により設計が複雑となる．情報漏洩のペイロードは，出力される信号線の値をトリガ信号に応じて指定の内部信号の値で上書きするものであり，マルチプレクサを用いることで単純な設計で構成できる．設計が単純な回路は通常回路との区別が難しく，HT 検知において検知されにくい．上記の理由から，提案手法によって生成された回路は，‘組合せ’ のトリガ回路と ‘情報流出’ のペイロード回路で構成されたと考えられる．

5. おわりに

本稿では，ランダムフォレストに基づく HT 検知に対し，強化学習を用いた自律的な HT 生成手法を提案した．提案手法で用いる強化学習アルゴリズムは，新たな HT の生成と HT 検知精度の観測を繰り返し，HT 検知において検知精度を最小化するよう，HT 構成時のパラメータ変更の方策を学習する．評価実験を通じて，提案手法により検知が困難な HT サンプルを生成することを確認した．今後の課題として，HT 検知精度をより低下させるような状態，行動，報酬の設計，および提案手法による HT 生成の効率性の検証が挙げられる．

謝辞 本研究成果は，国立研究開発法人情報通信研究機構の委託研究 (05201) により得られたものである．

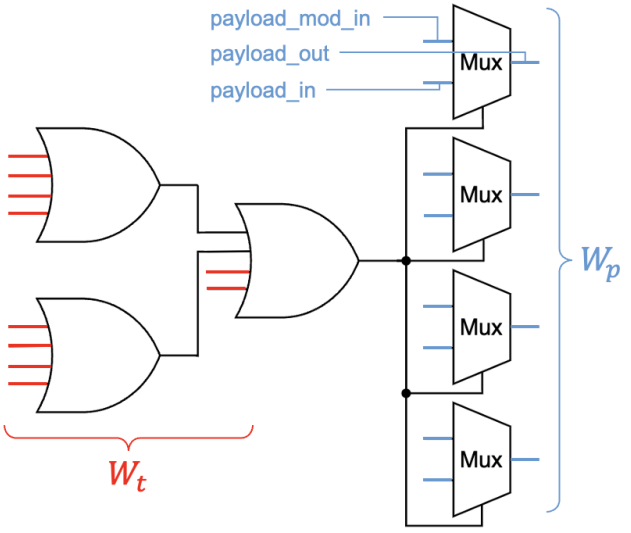

図 2 生成した HT のトリガ回路とペイロード回路の構成例. (トリガ回路のテンプレートが ‘組合せ’ で, パラメータとしてトリガ入力の信号線の数が 10, 発火条件となるトリガ入力の値が 0, ペイロード回路のテンプレートが ‘情報流出’ で, パラメータとして内部情報を含むペイロード入力 (payload_mod_in) と通常のペイロード入力 (payload_in), および出力 (payload_out) の信号線の数がそれぞれ 4 の場合)

参考文献

[1] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, “Hardware security: Threat models and metrics,” in *Proc. 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 819–823, 2013.

[2] J. Francq and F. Frick, “Introduction to hardware trojan detection methods,” in *Proc. 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 770–775, 2015.

[3] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, “Hardware trojans: Lessons learned after one decade of research,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 1–23, 2016.

[4] K. Hasegawa, M. Yanagisawa, and N. Togawa, “Empirical evaluation and optimization of hardware-trojan classification for gate-level netlists based on multi-layer neural networks,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 101, no. 12, pp. 2320–2326, 2018.

[5] K. Hasegawa, M. Yanagisawa, and N. Togawa, “Trojan-net feature extraction and its application to hardware-trojan detection for gate-level netlists using random forest,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 100, no. 12, pp. 2857–2868, 2017.

[6] K. Hasegawa, K. Yamashita, S. Hidano, K. Fukushima, K. Hashimoto, and N. Togawa, “Node-wise hardware trojan detection based on graph learning,” *arXiv preprint arXiv:2112.02213*, 2021.

[7] 根岸良太郎, 戸川望, “アンサンブル学習によるネットリストレベルのハードウェアトロイ識別の評価,” in *Proc. 2023 Symposium on Cryptography and Information Security (SCIS)*, 2023.

[8] 長谷川健人, 披田野清良, 福島和英, “グラフ学習にもとづく不正回路検知に対する強化学習を用いた自律的な脆弱性検査の提案,” in *Proc. 2022 Symposium on Cryptography and Information Security (SCIS)*, 2022.

[9] A. Sarihi, A. Patooghy, P. Jamieson, and A.-H. A. Badawy, “Trojan playground: A reinforcement learning framework for hardware trojan insertion and detection,” *arXiv preprint arXiv:2305.09592*, 2023.

[10] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[11] T. Kurihara and N. Togawa, “Hardware-trojan classification based on the structure of trigger circuits utilizing random forests,” in *Proc. 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–4, IEEE, 2021.

[12] 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, pp. 85–102, 2017.

[13] H. Salmani, M. Tehranipoor, and R. Karri, “On design vulnerability analysis and trust benchmarks development,” in *Proc. 2013 IEEE 31st international conference on computer design (ICCD)*, pp. 471–474, IEEE, 2013.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.