Appendix: Characterize the ability of GNNs in attacking logic locking

Omitted for Blind Review

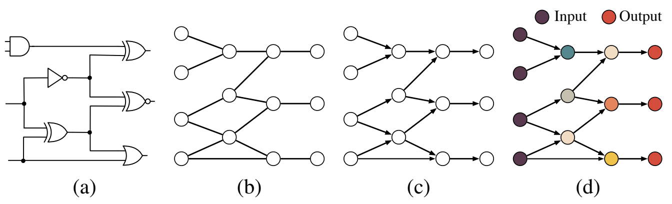

Fig. 1: (a) An example netlist with corresponding (b) undirected, (c) directed, and (d) heterogeneous graph models.

I. Preliminary

Graph representation of a logic netlist. In this paper, we focus on the gate-level logic netlist, which is composed of various gates, and interconnects. A netlist can be also modeled as a graph, where each gate is a node, and the interconnect form the edges. Differentiated by the graph type, there are three possible netlist graph representations:

- *Undirected netlist graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: each gate g_v corresponds to a node $v \in \mathcal{V}$, and $(u, v) \in \mathcal{E}$ if and only if g_v is the driver or the load of g_u.
- *Directed netlist graph* $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$: each gate g_v corresponds to a node $v \in \mathcal{V}'$, and $(u, v) \in \mathcal{E}'$ if and only if g_v is the driver of g_u.
- *Heterogeneous netlist graph* $\mathcal{G}^H = (\mathcal{V}^H = \cup \mathcal{V}_i : i \in$ *gates types*, $\mathcal{E}^H$): each gate g_v with type i corresponds to a node $v \in \mathcal{V}_i$, and $(u, v) \in \mathcal{E}^H$ if and only if g_v is the driver of g_u.

Especially, netlist inputs and outputs are also represented as nodes with types "input" and "output." Examples of three netlist graph representations are shown in Figure 1.

Weisfeiler-Lehman test. Weisfeiler-Lehman (WL) test [12] is a classical algorithm to determine whether two graphs are isomorphic, i.e., there exists a bijection $\pi : \mathcal{V} \to \mathcal{V}'$ between $\mathcal{G}_1$ and $\mathcal{G}_2$ such that $u, v \in \mathcal{V}$ are adjacent in $\mathcal{G}_1$ if and only if $\pi(u), \pi(v) \in \mathcal{V}'$ are adjacent in $\mathcal{G}_2$. [1] The 1-dimensional WL test is analogous to the iterative message-passing paradigm in GNNs, at each iteration, each node aggregates the labels from its neighborhoods, and then hashes the aggregated labels and its own label into a unique new label. The algorithm decides that two graphs are not isomorphic if the node labels between the two graphs differ at some iteration. Formally, each iteration t is defined as:

$$c_v^{(t)} = g\left(c_v^{(t-1)}, \left\{\left\{c_u^{(t-1)} : u \in N_v\right\}\right\}\right), \tag{1}$$

[1]The isomorphism of logic netlists are defined as mappings between gates rather than nodes.

where g denotes an injective hashing function, and $\{\{...\}\}$ represents a *multiset*, that is, a generalization of a set that allows repeated elements. Unless otherwise stated, the undirected WL test in the following context refers to the 1-dimensional WL test for undirected graphs. Similar to directed and heterogeneous GNNs, WL test can be extended to directed and heterogeneous case

Directed WL test. Similar to directed GNNs, the WL test is easily extended to a directed graph by aggregating successors and predecessors separately. That is,

$$c_v'^{(t)} = g\left(c_v'^{(t-1)}, \left\{\left\{c_s'^{(t-1)} : s \in S_v\right\}\right\}, \left\{\left\{c_p'^{(t-1)} : p \in P_v\right\}\right\}\right), \tag{2}$$

Heterogeneous WL test. The WL test is also extended to the heterogeneous cases where different metapaths give different multi-sets:

$$c_v^{H(t)} = g\left(c_v^{H(t-1)}, \left(\left\{\left\{c_u^{H(t-1)} : u \in \mathcal{N}_m(v)\right\}\right\} : m \in \text{metapaths}\right)\right), \tag{3}$$

II. Theoretical Framework

Definition 1 ($A \preceq B$)**.** *Assume GNNs and WL test have the same node attributes and initial node labels if the input graphs are from the same logic netlist. Given two logic netlists $\mathcal{N}_1, \mathcal{N}_2$, let the input of $A(B)$ be one of the three netlist graph representations, and defined as $\mathcal{G}_1^A, \mathcal{G}_2^A$ ($\mathcal{G}_1^B, \mathcal{G}_2^B$). We say that B is at least as powerful as A ($A \preceq B$) if and only if the following statement holds: If A decides $\mathcal{G}_1^A$ and $\mathcal{G}_2^A$ are not isomorphic, then B also decides $\mathcal{G}_1^B$ and $\mathcal{G}_2^B$ are not isomorphic.*

Specifically, we say $A = B$ if and only if $A \preceq B$ and $B \preceq A$, $A \prec B$ if and only if $A \preceq B$ holds but $B \preceq A$ does not hold. In the following section, we will construct the relationships among different GNNs and WL test variations based on the definition above.

A. Undirected vs. Directed vs. Heterogeneous

Lemma 1. *Undirected WL $\prec$ directed WL*

Proof. Define $\phi : \mathcal{V}_1 \cup \mathcal{V}_2 \to \mathcal{V}_1' \cup \mathcal{V}_2'$ as the mapping from nodes of the undirected graph $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ to the nodes of the corresponding directed graph $\mathcal{G}' = \mathcal{G}_1' \cup \mathcal{G}_2'$. We further define $\pi^t : \mathcal{V}_1 \cup \mathcal{V}_1' \to \mathcal{V}_2 \cup \mathcal{V}_2'$ as the isomorphism between $\bar{\mathcal{G}}_1 = \mathcal{G}_1 \cup \mathcal{G}_1'$ and $\bar{\mathcal{G}}_2 = \mathcal{G}_2 \cup \mathcal{G}_2'$ calculated by WL test and directed WL test after t iterations (if any). Here, $\cup$ between two graphs represents a super-graph containing each graph as an independent component.

Given any undirected graph and its corresponding directed graph with similar mapping definition, ϕ_*, π_*, we first show

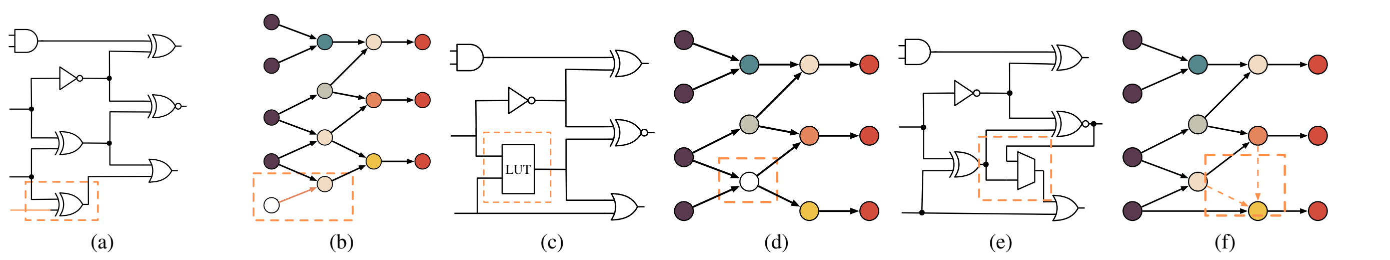

Fig. 2: Given the original logic netlist as in Figure 1a, the logic netlist is locked by (a) parity gate-based (c) LUT-based, and (e) MUX-based logic locking. The right graphs are their corresponding graph representations.

that if directed WL test labels two nodes u', v' the same after t iterations, i.e., $c'^{(t)}_{v'} = c'^{(t)}_{u'}$, the WL test always labels corresponding nodes u, v the same after t iterations, i.e., $c^{(t)}_v = c^{(t)}_u$, where $u' = \phi_*(u)$ and $v' = \phi_*(v)$. The statement holds for $t = 0$ by definition. Suppose this holds for iteration i, then, for $i+1$ iteration, if $c'^{(i+1)}_{v'} = c'^{(i+1)}_{u'}$, then, by definition, we have:

$$\begin{aligned} &\left(c'^{(i)}_{v'}, \left\{\left\{c'^{(i)}_s : s \in S_{v'}\right\}\right\}, \left\{\left\{c'^{(i)}_p : p \in P_{v'}\right\}\right\}\right) \quad (4)\\ =& \left(c'^{(i)}_{u'}, \left\{\left\{c'^{(i)}_s : s \in S_{u'}\right\}\right\}, \left\{\left\{c'^{(i)}_p : p \in P_{u'}\right\}\right\}\right) \end{aligned}$$

Let $N_{v'}$ be the neighborhood set of node v', i.e., $\mathcal{N}'_{v'} = S_{v'} \cup P_{v'}$. Then, it must the case that:

$$\left(c'^{(i)}_{v'}, \left\{\left\{c'^{(i)}_n : n \in \mathcal{N}_{v'}\right\}\right\}\right) = \left(c'^{(i)}_{u'}, \left\{\left\{c'^{(i)}_n : n \in \mathcal{N}_{u'}\right\}\right\}\right) \quad (5)$$

By the definition of the relationship between undirected graph and its corresponding directed graph, we have $\mathcal{N}_{v'} = \mathcal{N}_{\phi_*(v)} = \phi_*(\mathcal{N}_v)$. Replace $\mathcal{N}_{v'}$ and $\mathcal{N}_{u'}$ in Eq 5 by $\phi_*(\mathcal{N}_v)$ and $\phi_*(\mathcal{N}_u)$, we have:

$$\begin{gathered} \left\{\left\{c'^{(i)}_{\phi_*(n)} : n \in \mathcal{N}_v\right\}\right\} = \left\{\left\{c'^{(i)}_{\phi_*(n)} : n \in \mathcal{N}_u\right\}\right\} \quad (6)\\ c'^{(i)}_{v'} = c'^{(i)}_{u'} \end{gathered}$$

Then, by our assumption on iteration i, we must have:

$$\left(c^{(i)}_v, \left\{\left\{c^{(i)}_n : n \in \mathcal{N}_v\right\}\right\}\right) = \left(c^{(i)}_u, \left\{\left\{c^{(i)}_n : n \in \mathcal{N}_u\right\}\right\}\right) \quad (7)$$

Therefore, by induction, if directed WL test labels two nodes u', v' the same, the WL test always labels corresponding nodes u, v the same. This means that there always exists a mapping σ such that $c^{(i)}_v = m\left(c'^{(i)}_{\phi(v)}\right)$ for any $v \in \mathcal{G}$ and $i \in \mathbb{N}$.

Suppose after k iterations, the WL test decides $\mathcal{G}_1$ and $\mathcal{G}_2$ are not isomorphic, that is,

$$\left\{\left\{c^{(k)}_v : v \in \mathcal{G}_1\right\}\right\} \neq \left\{\left\{c^{(k)}_{\pi(v)} : \pi(v) \in \mathcal{G}_2\right\}\right\} \quad \forall \pi : \bar{\mathcal{G}}_1 \to \bar{\mathcal{G}}_2 \quad (8)$$

Combining with our previous proof that $c^{(i)}_v = m\left(c'^{(i)}_{\phi(v)}\right)$, we have:

$$\left\{\left\{c'^{(k)}_{\phi(v)} : \phi(v) \in \mathcal{G}'_1\right\}\right\} \neq \left\{\left\{c'^{(k)}_{\phi(\pi(v))} : \phi(\pi(v)) \in \mathcal{G}'_2\right\}\right\} \quad (9)$$

$$\forall \pi : \bar{\mathcal{G}}_1 \to \bar{\mathcal{G}}_2$$

which means that there is no isomorphism mapping between $\mathcal{G}'_1$ and $\mathcal{G}'_2$. In other words, the directed WL test also decides the non-isomorphism of $\mathcal{G}'_1$ and $\mathcal{G}'_2$, hence we reached a contradiction, and showed that undirected WL $\preceq$ directed WL.

We then deny the possibility of them being equivalently powerful: a simple example is given in Figure 3, where the two different logic netlists give the same undirected graph representation. In the example, the undirected WL test cannot distinguish the logic netlists given the same undirected netlist graph, but the directed WL test can distinguish them even after only one iteration.

□

Besides undirected and directed WL test on the netlist graphs, we can also derive the relationship between the directed WL test and heterogeneous WL test:

Lemma 2. *Directed WL $\preceq$ heterogeneous WL*

Proof. We first derive that if heterogeneous WL test labels two nodes u^H, v^H the same after t iterations, the directed WL test always labels corresponding nodes u', v' the same after t iterations. The key is that the multisets at each iteration in the heterogeneous netlist graph can be reduced to the multisets of the directed WL test: $\cup \mathcal{N}_{m^+}(v^H) : m^+ \in \{i\text{−}\mathtt{POS}\text{−}type(v) : i \in \text{types of nodes}\} = \phi(P(v'))$ and $\cup \mathcal{N}_{m^-}(v^H) : m^- \in \{i - \mathtt{NEG} - type(v) : i \in \text{types of nodes}\} = \phi(S(v'))$ Then, we can show that heterogeneous WL test always decides the non-isomorphism of $\mathcal{G}^H_1$ and $\mathcal{G}^H_2$ as long as the directed WL test decides the non-isomorphism of $\mathcal{G}'_1$ and $\mathcal{G}'_2$ since there exists a mapping such that $c'^{(i)}_v = m\left(c^{H,(i)}_{\phi(v)}\right)$. □

Unlike the comparison between directed and undirected WL test, it is possible for directed WL test to be as powerful as the heterogeneous WL test. The conditions are specified as follows:

Lemma 3. *Directed WL = heterogeneous WL when nodes with different gate types are assigned different initial node labels.*

Proof. We follow the definitions in lemma 1 with the “undirected” term is replaced by “heterogeneous”. When different gates have different initial node labels in the directed WL test, it clearly holds that their node labels are always different after any number of iterations of directed WL test.

We first show that if such a directed WL test labels two nodes u', v' the same after t iteration, the heterogeneous WL test

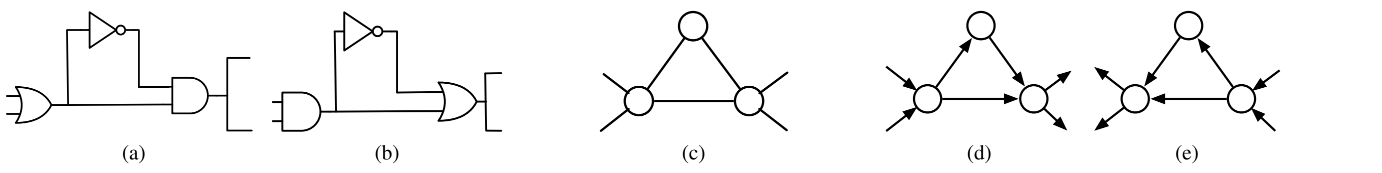

Fig. 3: (a) (b) Two different logic netlists; (c) corresponding undirected graph representation; (d) (e) corresponding directed graph representations

will also label corresponding nodes u^H, v^H the same. The statement holds for $t = 0$ by definition. Suppose this holds for iteration i, then, for $i+1$ iteration, if $c'^{(i+1)}_{v'} = c'^{(i+1)}_{u'}$, then, by definition, we have:

$$\begin{aligned}&\left(c'^{(i)}_{v'}, \left\{\left\{c'^{(i)}_s : s \in S_{v'}\right\}\right\}, \left\{\left\{c'^{(i)}_p : p \in P_{v'}\right\}\right\}\right) \\ &= \left(c'^{(i)}_{u'}, \left\{\left\{c'^{(i)}_s : s \in S_{u'}\right\}\right\}, \left\{\left\{c'^{(i)}_p : p \in P_{u'}\right\}\right\}\right)\end{aligned} \tag{10}$$

Since nodes with different gate types have different node labels, that means: $c'_{v'} \neq c'_{u'}$ for all $v' \in \mathcal{V}'_1, u' \in \mathcal{V}'_2$ if the gates types of v' and u' are different. This gives the following:

$$\begin{aligned}\left\{\left\{c'^{(i)}_n : n \in \mathcal{N}_{a-e}(v')\right\}\right\} \neq \left\{\left\{c'^{(i)}_n : n \in \mathcal{N}_{b-e}(u')\right\}\right\} \\ : a \neq b, e \in \{POS, NEG\}\end{aligned} \tag{11}$$

where $\mathcal{N}_{a-POS}(v)\,(\mathcal{N}_{a-NEG}(v'))$ represents the predecessors (successors) of v with gate type a. Combing with Equation (10) and Equation (11), if $c'^{(i+1)}_{v'} = c'^{(i+1)}_{u'}$, we must have:

$$\begin{aligned}\left\{\left\{c'^{(i)}_n : n \in \mathcal{N}_{a-e}(v')\right\}\right\} = \left\{\left\{c'^{(i)}_n : n \in \mathcal{N}_{a-e}(u')\right\}\right\} \\ : e \in \{POS, NEG\}\end{aligned} \tag{12}$$

Therefore, by induction, if directed WL test labels two nodes u', v' the same, the heterogeneous WL test always labels corresponding nodes u^H, v^H the same. This means that there always exists a mapping σ such that $c^{H(i)}_{v^H} = m\left(c'^{(i)}_{v'}\right)$. We can then show that the directed WL test always decides the non-isomorphism of $\mathcal{G}'_1$ and $\mathcal{G}'_2$ as long as the heterogeneous WL test decides the non-isomorphism of $\mathcal{G}^H_1$ and $\mathcal{G}^H_2$.[2] $\square$

We already show that the heterogeneous WL test is the most powerful among three WL test variations. The inequality relationships also hold in GNNs: It is easy to see that heterogeneous GNNs are at least as powerful as other GNN variations, since the heterogeneous GNNs can always reduce to directed GNNs and normal GNNs. Formally, we have:

Lemma 4. *Undirected GNNs $\prec$ directed GNNs*

Lemma 5. *Directed GNNs $\preceq$ heterogeneous GNN*

Proof. We only show the proof of Lemma 4 due to the page limit, the proof of Lemma 5 is similar. We finish the proof by showing that any undirected GNN can be obtained by downgrading some directed GNN. Define $\mathcal{A} = \left\{\left(\text{AGG}^{(i)}, \text{COM}^{(i)}\right)_{i=1}^{L}, \text{READ}\right\}$ maps $\mathcal{G}_1$ and $\mathcal{G}_2$ to different embeddings. Then a directed GNN $\mathcal{A}' = \left\{\left(\text{AGG}'^{(i)}, \text{COM}'^{(i)}\right)_{i=1}^{L}, \text{READ}'\right\}$ can be constructed by:

$$\text{AGG}'^{(i)}(\boldsymbol{c}, \mathbb{S}, \mathbb{P}) = \text{AGG}^{(i)}(\boldsymbol{c}, (\mathbb{S} \cup \mathbb{P})) \quad \forall i \in \{1, \ldots, L\} \tag{13}$$

$$\text{COM}'^{(i)}(\boldsymbol{x}, \boldsymbol{y}) = \text{COM}^{(i)}(\boldsymbol{x}, \boldsymbol{y}) \quad \forall i \in \{1, \ldots, L\} \tag{14}$$

$$\text{READ}'(\mathbb{X}) = \text{READ}(\mathbb{X}) \tag{15}$$

Since $\mathcal{G}$ and $\mathcal{G}'$ have the same initial node features, i.e., $\boldsymbol{h}^0_v = \boldsymbol{h}^0_{\phi(v)}$, and $\mathcal{S}'_v \cup \mathcal{P}'_v = \mathcal{N}_v$ by definition, $\mathcal{A}$ and $\mathcal{A}'$ always generate the same node embeddings at each layer, thereby always generating the same graph embedding. Therefore, we show that undirected GNNs $\preceq$ directed GNNs. Moreover, undirected GNNs cannot be as powerful as directed GNNs: one of the examples is given in Figure 3, where the undirected GNNs cannot distinguish the two logic netlists since the given undirected graph representations are totally the same. $\square$

B. GNNs versus WL test

Xu [10] demonstrates that the power of GNNs is bounded by WL test in determining the isomorphism of undirected graphs. However, as shown in previous lemmas, both GNNs and WL test are not as powerful as their directed and heterogeneous versions in determining the isomorphism of netlists. Here, we compare GNNs and WL test by directly comparing their maximally powerful version: the heterogeneous GNNs and heterogeneous WL test:

Lemma 6. *Heterogeneous GNNs $\preceq$ heterogeneous WL test*

Proof. We first show that if the heterogeneous WL test labels two nodes the same after t iterations, i.e., $c^{(t)}_v = c^{(t)}_u$, then heterogeneous GNNs will always generate the same node features at t_{th} iteration, i.e., $\boldsymbol{h}^{(t)}_v = \boldsymbol{h}^{(t)}_u$. Suppose this holds for iteration i, then, for $i+1$ iteration, if $c^{(i+1)}_v = c^{(i+1)}_u$, then, by definition, we have:

$$\begin{aligned}&\left(c^{(i)}_v, \left\{\left\{c^{(i)}_n : n \in \mathcal{N}_m(v)\right\}\right\} : m \in \text{metapaths}\right) = \\ &\left(c^{(i)}_u, \left\{\left\{c^{(i)}_n : n \in \mathcal{N}_m(u)\right\}\right\} : m \in \text{metapaths}\right)\end{aligned} \tag{16}$$

Since $c^{(i)}_v = c^{(i)}_u$ indicates $\boldsymbol{h}^{(i)}_v = \boldsymbol{h}^{(i)}_u$ by our assumption on iteration i, it must be the case that

$$\begin{aligned}&\left(\boldsymbol{h}^{(i)}_v, \left\{\left\{\boldsymbol{h}^{(i)}_n : n \in \mathcal{N}_m(v)\right\}\right\} : m \in \text{metapaths}\right) = \\ &\left(\boldsymbol{h}^{(i)}_u, \left\{\left\{\boldsymbol{h}^{(i)}_n : n \in \mathcal{N}_m(u)\right\}\right\} : m \in \text{metapaths}\right)\end{aligned} \tag{17}$$

Therefore, by induction, if heterogeneous WL test labels two nodes the same, the heterogeneous GNNs will always

[2]The proof is similar to the proof in Lemma 1 and not shown here.

generate the same node features. This means that there always exists a mapping σ such that $\boldsymbol{h}_v^{(i)} = m\left(c_v^{(i)}\right)$. Then, for two heterogeneous netlist graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, if heterogeneous WL test fails to determine the isomorphism of the two graphs, i.e., $\{\{c_v : v \in \mathcal{V}_1\}\} = \{\{c_v : v \in \mathcal{V}_2\}\}$, the multisets of node features must be the same:

$$\{\{\boldsymbol{h}_v : v \in \mathcal{V}_1\}\} = \{\{\boldsymbol{h}_v : v \in \mathcal{V}_2\}\} \tag{18}$$

Because the readout function is permutation invariant, we have $A(\mathcal{G}_1) = A(\mathcal{G}_2)$, which is a contradiction. □

Although GNNs are upper bounded by heterogeneous WL test, they are still possible to reach the bound, i.e., be equally powerful as the heterogeneous WL test:

Theorem 1. *Let $\mathcal{A}$ be a GNN with a sufficient number of GNN layers. $\mathcal{A}$ = heterogeneous WL test if the following conditions hold:*

- *the aggregation and combination functions of $\mathcal{A}$ are injective.*
- *the readout function of $\mathcal{A}$ is injective.*
- *At least one of the two conditions holds: 1) $\mathcal{A}$ separately aggregates features from different meta-paths. 2) $\mathcal{A}$ separately aggregates features from the successors and predecessors, and nodes with different gate types are assigned different initial node attributes.*

The summary of all relationships are given in Figure 4.

Proof. We first consider the case where the first sub-condition in the third condition holds: Let $\mathcal{A}$ be the GNN satisfying the conditions above, $\boldsymbol{h}_v^{(i)}$ be the node feature of v generated by $\mathcal{A}$ at iteration i, and $c_v^{(i)}$ be the node label of v assigned by the heterogeneous GNN at iteration i. Assume that the heterogeneous WL test decides two heterogeneous netlist graphs $\mathcal{G}_1, \mathcal{G}_2$ as non-isomorphic after t iteration, but $\mathcal{A}$ still fails to decide the isomorphism.

We first show that there always exists an injective function σ such that $\boldsymbol{h}_v^{(i)} = \sigma\left(c_u^{(i)}\right)$ for all $i \in \{1, ..., k\}$. The statement holds for $k = 0$ since the node attributes and initial node labels are the same. Suppose this holds for iteration i, that is, $\boldsymbol{h}_v^{(i)} = \sigma\left(c_u^{(i)}\right)$, then, $\boldsymbol{h}_v^{(i+1)}$ will be:

$$\boldsymbol{h}_v^{(i+1)} = \alpha\left(\sigma\left(c_v^{(i)}\right), \beta\left(\left\{\sigma\left(c_u^{(i)}\right) : u \in \mathcal{N}_m(v)\right\} : m \in \text{metapaths}\right)\right) \tag{19}$$

where α, β are corresponding injective combination and aggregation functions. Because the composition of injective functions is also injective, the equation above can be further represented by:

$$\boldsymbol{h}_v^{(i+1)} = \epsilon\left(c_v^{(i)}, \left(\left\{c_u^{(i)} : u \in \mathcal{N}_m(v)\right\} : m \in \text{metapaths}\right)\right), \tag{20}$$

where ϵ is an injective function. Then, we have:

$$\begin{aligned}\boldsymbol{h}_v^{(i+1)} &= \epsilon \circ g^{-1} \circ g\left(c_v^{(i)}, \left(\left\{c_u^{(i)} : u \in \mathcal{N}_m(v)\right\} : m \in \text{metapaths}\right)\right) \quad (21)\\ &= \epsilon \circ g^{-1}\left(c_v^{(i+1)}\right)\end{aligned}$$

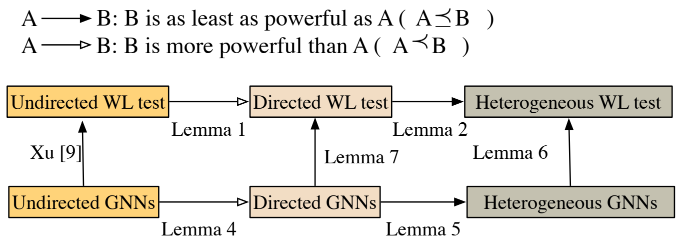

Fig. 4: The power relations of GNNs and WL tests in determining the logic netlist isomorphism. Different colors represent different graph representations.

Since both ϵ and g^{-1} are injective, there always exists an injective function that maps $c_v^{(i)}$ to $\boldsymbol{h}_v^{(i)}$. According to our assumption that $\mathcal{A}$ fails to decide the isomorphism and the readout function is injective, we have:

$$\left\{\boldsymbol{h}_v^{(t)} : v \in \mathcal{G}_1\right\} = \left\{\boldsymbol{h}_u^{(t)} : u \in \mathcal{G}_2\right\} \tag{22}$$

That is,

$$\left\{\sigma\left(c_v^{(t)}\right) : v \in \mathcal{G}_1\right\} = \left\{\sigma\left(c_u^{(t)}\right) : u \in \mathcal{G}_2\right\} \tag{23}$$

which is a contradiction. Therefore, we prove that such a $\mathcal{A}$ always decides the non-isomorphism as long as the heterogeneous WL test determines the two graphs as non-isomorphic.

We then consider $\mathcal{A}$ being the GNN satisfying the second sub-condition and other two conditions, i.e., the aggregation, combination, and readout functions are injective, $\mathcal{A}$ separately aggregates features from the successors and predecessors, and nodes with different gate types are assigned different initial node attributes. Similarly, we can show that:

Lemma 7. *Let $\mathcal{A}$ be a GNN with a sufficient number of GNN layers. $\mathcal{A}$ = directed WL test if 1) the aggregation, combination, and readout functions are injective. 2) $\mathcal{A}$ separately aggregates features from the successors and predecessors.*

Due to the page limit, we do not show the proof here. Combining with the fact that directed WL test = heterogeneous WL test if and only if the initial node labels of directed WL test are different based on the gate types, we can conclude that GNN = heterogeneous WL test only when conditions in Lemma 7 hold and the node attributes of $\mathcal{A}$ are different based on the gate types. Therefore, we finish the proof of the second case. □

C. Constructing powerful GNNs

Based on Theorem 1, a guidance for building powerful GNNs can be naturally provided. However, the theoretical upper bound is hard to reach since injectivity cannot be guaranteed, and the requirement of layers being sufficiently deep is not feasible in practice. Moreover, injective functions are not unique, meaning that significant efforts are still needed to determine which GNN setting is better. In this section, we further explore the influence of some specific GNN settings on the attacking problem.

Injective function Xu [10] shows that sum aggregators over the multiset can be injective. that is, there exists a function

f so that $h(X) = \sum_{x \in X} f(x)$ is unique for each multiset X. However, both combination and aggregation functions of directed and heterogeneous GNNs receive *ordered multi-sets*, rather than single multi-set. Here, we extend the injective functions to the case of multiple ordered inputs:

Lemma 8 (injective over ordered inputs)**.** *Assume $\mathcal{X}$ is countable. There exists a function pair (f, f') so that $h(X,Y) = \sum_{x \in X} f(x) + \sum_{y \in Y} f'(y)$ is unique for each ordered pair (X, Y), where $X \in \mathcal{X}$ and $Y \in \mathcal{X}$ are two multisets of bounded size.*

Proof. Because $\mathcal{X}$ is countable, there always exists a mapping $Z : \mathcal{X} \to \mathbb{N}$. At the same time, there exists a number $N \in \mathbb{N}$ so that $|X| < N$ for all X since the cardinality of multisets X is bounded. Then, $f(x) = (2N)^{(-Z(x))}, f'(x) = (2N)^{(-Z(x)-N)}$ satisfies the requirement. □

The Lemma above can be extended from the case of two inputs to more inputs: functions can be injective as long as different inputs are processed by different functions (like f and f' here) separate.

Attention Among various possible options for injective functions, we may choose the one that is able to generate more diverse graph embeddings. Intuitively, the more diverse the results are, the more possible it is for the two different logic netlists to be distinguished. To make the final embeddings of different logic netlists as distinct as possible, we adopt the attention mechanism. In GNNs, different neighbors are assigned different attentions that based on weighted sum. Here, we first show that the weighted sum over neighbors indeed makes the final embeddings more diverse:

Lemma 9 (attention)**.** *Assume $x \in \mathcal{X}$ is the node feature drawn from the same distribution with mean μ_0 and variance σ_0^2, and the node features are i.i.d. The coefficient of variance of x's weighted sum is always larger than or equal to the coefficient of variance of x's sum/mean, i.e., $CV\left(\sum_{x \in \mathcal{X}} \epsilon_x x\right) \geq CV\left(\sum_{x \in \mathcal{X}} x\right)$ and $CV\left(\sum_{x \in \mathcal{X}} \epsilon_x x\right) \geq CV\left(\frac{\sum_{x \in \mathcal{X}} x}{|\mathcal{X}|}\right)$, where $\sum_{x \in \mathcal{X}} \epsilon_x = 1$.*

Proof. Define $x'_{\text{att}} = \sum_{x \in \mathcal{X}} \epsilon_x x$, $x'_{\text{sum}} = \sum_{x \in \mathcal{X}} x$, $x'_{\text{mean}} = \frac{\sum_{x \in \mathcal{X}} x}{|\mathcal{X}|}$. The coefficient of variance of x'_{att}, x'_{sum} and x'_{mean} are computed by:

$$CV\left(x'_{\text{att}}\right) = \frac{\sqrt{var(x'_{\text{att}})}}{E(x'_{\text{att}})} = \frac{\sqrt{\left(\sum_{x \in \mathcal{X}} \epsilon_x^2\right) \cdot \sigma_0^2}}{\left(\sum_{x \in \mathcal{X}} \epsilon_x\right) \cdot \mu_0} \tag{24}$$

$$= \frac{\sqrt{\left(\sum_{x \in \mathcal{X}} \epsilon_x^2\right)} \cdot \sigma_0}{\cdot \mu_0} \tag{25}$$

$$CV\left(x'_{\text{sum}}\right) = \frac{\sqrt{var(x'_{\text{sum}})}}{E(x'_{\text{sum}})} = \frac{\sqrt{|\mathcal{X}| \cdot \sigma_0^2}}{|\mathcal{X}| \cdot \mu_0} = \frac{\sigma_0}{\sqrt{|\mathcal{X}|} \cdot \mu_0} \tag{26}$$

$$CV\left(x'_{\text{mean}}\right) = \frac{\sqrt{var(x'_{\text{mean}})}}{E(x'_{\text{mean}})} = \frac{\sqrt{\frac{\sigma_0^2}{|\mathcal{X}|}}}{\mu_0} = \frac{\sigma_0}{\sqrt{|\mathcal{X}|} \cdot \mu_0} \tag{27}$$

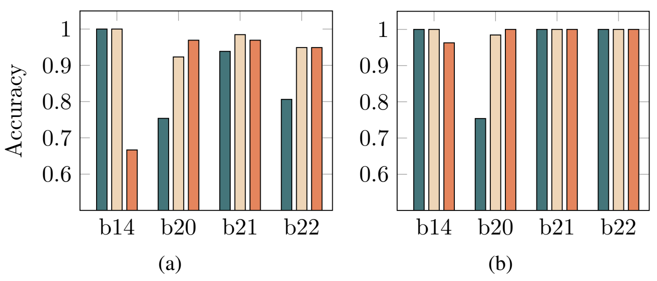

Fig. 5: The (a) top-1 accuracy and (b) top-3 accuracy in attacking the LUT-based method. The green bar represents graphSAGE [14], the light orange bar represents heterogeneous NetlistGNN, and the dark orange bar represents directed NetlistGNN.

By the Cauchy-Schwarz inequality:

$$1 = \left(\sum_{x \in \mathcal{X}} \epsilon_x\right)^2 = \left(\sum_{x \in \mathcal{X}} 1 \cdot \epsilon_x\right)^2 \leq |\mathcal{X}| \cdot \left(\sum_{x \in \mathcal{X}} \epsilon_x^2\right) \tag{28}$$

Combining the equations above, we have:

$$CV\left(x'_{\text{att}}\right) = \frac{\sqrt{\left(\sum_{x \in \mathcal{X}} \epsilon_x^2\right)} \cdot \sigma_0}{\cdot \mu_0} \geq \frac{\sigma_0}{\sqrt{|\mathcal{X}|} \cdot \mu_0} \tag{29}$$

This completes the proof. □

Depth Theorem 1 assumes a sufficient number of layers, which are not feasible. But, it is still interesting to explore the relationship between depth and GNN's power in logic locking. We observe the following:

Lemma 10 (Depth)**.** *Let k-$\mathcal{A}$ be the GNN with depth k, and all aggregation, combination, and readout functions being injective, then we have k-$\mathcal{A} \leq (k+1)$-$\mathcal{A}$*

The proof is obvious: if k-$\mathcal{A}$ maps two graphs to different graph embeddings, then $(k+1)$-$\mathcal{A}$ also always maps them to different graph embeddings since the functions are injective.

From the above lemma, we can see that increasing the depth of GNNs improves its theoretical upper bound. However, too deep network makes the model hard to generalize to new data, and difficult to be trained due to the over-smoothing issue.

III. Results

Comparison with other GNN models on LUT-based logic lock. We compare graphSAGE on the LUT-based logic lock using the ITC-99 dataset, as shown in Figure 5. Generally, NetlistGNN outperforms GNNUnlock on both top-1 and top-3 accuracy. Interestingly, for the small logic netlist (b14), the attention mechanism harms the performance due to the over-fitting in the small datasets.

Attention In Table I, "Ours w.o. attention" is the directed NetlistGNN without attention module; It is observed that the attention-based directed model (row "Ours directed") is better than the heterogeneous model (row "Ours heterogeneous") and without-attention variant (row "Ours w.o. attention") in larger cases (sha256, b18, b19), but is worse in small cases. The reason lies in the model complexity: attention-based GNNs are

TABLE I: Prediction accuracy of GraphSAGE [14], GATv2 [19], and NetlistGNN on Truly Random Logic lock [4]

Dataset	ISCAS85		CEP					ITC99						
Benchmark	c5315	c7552	des3	sha256	FIR_filter	md5	IIR_filter	b14	b18	b19	b20	b21	b22
# of nodes	3968	4302	13128	53385	16225	21766	26091	8022	114463	223111	20022	20411	29147
GraphSAGE	0.8311	0.8766	0.834	0.9793	0.992	0.9175	0.9692	0.8277	0.8871	0.8754	0.8655	0.8647	0.8659
Ours directed	0.8378	0.9481	0.9538	**0.9948**	0.994	0.9782	0.9835	**0.8986**	**0.9687**	**0.9834**	**0.9556**	0.9499	0.9577
Ours heterogeneous	**0.9527**	**0.9675**	**0.9769**	0.9868	**1.0**	**0.9794**	**0.9956**	**0.8986**	0.9627	0.9739	0.9417	**0.9513**	**0.9622**
Ours w. undirected (GATv2)	0.8649	0.8247	0.8803	0.9807	0.976	0.921	0.9714	0.8243	0.9217	0.9157	0.8308	0.866	0.9415
Ours w.o. attention	0.8851	0.9545	0.9622	0.9939	0.992	0.9817	0.9923	0.9122	0.963	0.9819	0.9473	0.9526	0.964
Ours-2	0.7542	0.7727	0.8277	0.8614	0.8842	0.8889	0.8282	0.7027	0.7001	0.6942	0.7614	0.7145	0.8137
Ours-4	0.8581	0.9459	0.9622	0.9915	0.994	0.9817	0.9857	0.8784	0.9671	0.9639	0.9459	0.9472	0.9469
Ours-16	0.8514	0.9091	0.9433	0.991	0.984	0.9851	0.9747	0.8851	0.9592	0.972	0.939	0.9445	0.9505

more complex to train the attention weights, but small cases are lack of enough labeled nodes for attention-based GNNs, making it hard to be generalized well.

Depth In Table I, "Ours-k" is the directed NetlistGNN with k layers (model depth). When the model depth is not large, the accuracy increases with the depth, which aligns with our analysis. However, when the model contains 16 layers, the accuracy even drops. We attribute the decline to the model complexity: deeper model needs more data to generalize. In other words, deeper model may be more suitable for larger circuits: As shown in Table I, Ours-16 has a similar performance on large benchmarks, but the accuracy drops a lot on small benchmarks.

References

[1] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[2] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," *Design, Automation & Test in Europe*, pp. 1069–1074, 2008.

[3] H. M. Kamali, K. Z. Azar, K. Gaj, H. Homayoun, and A. Sasan, "LUT-lock: A Novel LUT-Based Logic Obfuscation for FPGA-Bitstream and ASIC-Hardware Protection," *IEEE Computer Society Annual Symposium on VLSI*, pp. 405–410, 2018.

[4] N. Limaye, E. Kalligeros, N. Karousos, I. G. Karybali, and O. Sinanoglu, "Thwarting all logic locking attacks: Dishonest oracle with truly random logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1740–1753, 2020.

[5] D. Sisejkovic, F. Merchant, L. M. Reimann, and R. Leupers, "Deceptive logic locking for hardware integrity protection against machine learning attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[6] A. Alaql, M. M. Rahman, and S. Bhunia, "Scope: Synthesis-based constant propagation attack on logic locking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 8, pp. 1529–1542, 2021.

[7] L. Alrahis, S. Patnaik, M. Shafique, and O. Sinanoglu, "Muxlink: Circumventing learning-resilient mux-locking using graph neural network-based link prediction," *arXiv preprint arXiv:2112.07178*, 2021.

[8] L. Alrahis, S. Patnaik, M. A. Hanif, M. Shafique, and O. Sinanoglu, "Untangle: Unlocking routing and logic obfuscation using graph neural networks-based link prediction," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[9] L. Alrahis, S. Patnaik, F. Khalid, M. A. Hanif, H. Saleh, M. Shafique, and O. Sinanoglu, "Gnnunlock: Graph neural networks-based oracle-less unlocking scheme for provably secure logic locking," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 780–785.

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[11] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," vol. 33, 2019, pp. 4602–4609.

[12] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[14] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017, pp. 1024–1034.

[15] J. Sweeney, R. Purdy, R. D. Blanton, and L. Pileggi, "Circuitgraph: A python package for boolean circuits," *Journal of Open Source Software*, vol. 5, no. 56, p. 2646, 2020.

[16] L. Alrahis, A. Sengupta, J. Knechtel, S. Patnaik, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "Gnn-re: Graph neural networks for reverse engineering of gate-level logic netlists," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[17] Z. He, Z. Wang, C. Bail, H. Yang, and B. Yu, "Graph learning-based arithmetic block identification," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–8.

[18] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction," 2022.

[19] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[20] L. Alrahis, S. Patnaik, M. Shafique, and O. Sinanoglu, "Embracing graph neural networks for hardware security," *arXiv preprint arXiv:2208.08554*, 2022.