Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation

Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V. Tripunitara,
University of Waterloo

This paper is included in the Proceedings of the 22nd USENIX Security Symposium.

August 14–16, 2013 • Washington, D.C., USA

ISBN 978-1-931971-03-4

Open access to the Proceedings of the 22nd USENIX Security Symposium is sponsored by USENIX

Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation

Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V. Tripunitara

ECE, University of Waterloo, Canada
{*fcimeson,aemtenan,siddharth.garg,tripunit*}*@uwaterloo.ca*

Abstract

The fabrication of digital Integrated Circuits (ICs) is increasingly outsourced. Given this trend, security is recognized as an important issue. The threat agent is an attacker at the IC foundry that has information about the circuit and inserts covert, malicious circuitry. The use of 3D IC technology has been suggested as a possible technique to counter this threat. However, to our knowledge, there is no prior work on how such technology can be used effectively. We propose a way to use 3D IC technology for security in this context. Specifically, we obfuscate the circuit by lifting wires to a trusted tier, which is fabricated separately. This is referred to as split manufacturing. For this setting, we provide a precise notion of security, that we call *k*-security, and a characterization of the underlying computational problems and their complexity. We further propose a concrete approach for identifying sets of wires to be lifted, and the corresponding security they provide. We conclude with a comprehensive empirical assessment with benchmark circuits that highlights the security versus cost trade-offs introduced by 3D IC based circuit obfuscation.

1 Introduction

The security of digital integrated circuits (ICs), the building blocks of modern computer hardware systems, can be compromised by covertly inserted malicious circuits. The threat from such maliciously inserted hardware is of increasing concern to government and military agencies [2] and commercial semiconductor vendors. Recently, Skorobogatov et al. [28] demonstrated the presence of a backdoor in a military grade FPGA manufactured by Actel that enabled access to configuration data on the chip. The authors initially conjectured that the backdoor was maliciously inserted since the key used to trigger the backdoor was undocumented. Actel has since clarified that the backdoor was inserted by design for internal test purposes [23]. Nonetheless, this incident has further heightened the perceived threat from maliciously inserted hardware, and effective counter-measures to deter or prevent such attacks are of increasing importance.

The threat of maliciously inserted hardware arises from two factors. First, owing to their complexity, digital ICs are designed at sites across the world. In addition, parts of the design are often outsourced or purchased from external vendors. Second, a majority of semiconductor design companies are fabless, i.e., they outsource IC manufacturing to a potentially untrusted external fabrication facility (or foundry). Both factors make it easier for a malicious attacker in a design team or a malicious foundry (or a collusion between the two) to insert covert circuitry in a digital IC.

Three-dimensional (3D) integration, an emerging IC manufacturing technology, is a promising technique to enhance the security of computer hardware. A 3D IC consists of two or more independently manufactured ICs that are vertically stacked on top of each other — each IC in the stack is referred to as a *tier*. Interconnections between the tiers are accomplished using vertical metal pillars referred to as through-silicon vias (TSV).

3D IC manufacturing can potentially enhance hardware security since each tier can be manufactured in a separate IC foundry, and vertically stacked in a secure facility. Thus, a malicious attacker at any one foundry has an incomplete view of the entire circuit, reducing the attacker's ability to alter the circuit functionality in a desired manner.

Tezarron, a leading commercial provider of 3D stacking capabilities, has alluded to the enhanced security offered by 3D integration in a white paper [1]. The white paper notes that "A multi-layer circuit may be divided among the layers in such a way that the function of each layer becomes obscure. Assuming that the TSV connections are extremely fine and abundant, elements can be scattered among the layers in apparently random fashion." However, the paper does not provide any formal

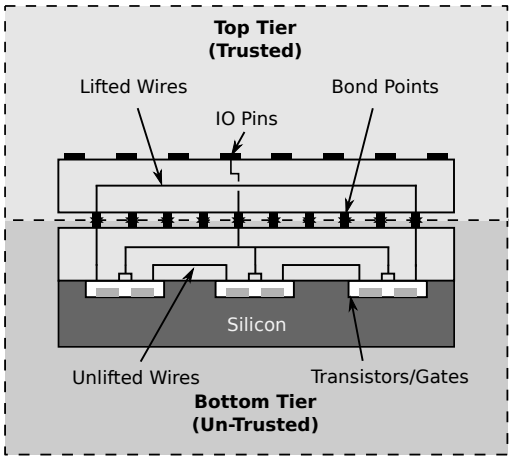


Figure 1: A two tier 3D IC. In this instance, the top tier is an interposer, i.e., it only implements metal wires, while the bottom tier has both transistors/gates and wires.

notion of security for split manufacturing, nor does it propose techniques to quantify security or achieve a certain security level. These are the open challenges that we address in this paper.

Our threat model assumes a malicious attacker in an IC foundry who wants to modify the functionality of a digital IC in a specific, targeted manner. The attack proposed by King et al. [19] modifies the state of hardware registers in a processor to raise the privilege level of the attacker — this is an example of a targeted attack since it requires the attacker to determine the gate or wire in the circuit that corresponds to the privilege bit. Fault insertion attacks in cryptographic hardware also require that certain vulnerable bits be targeted. For example, it has been shown that if the LSB bit of the 14^{th} round of a DES implementation is set to logic zero, the secret key can be recovered in as few as two messages [9]. However, to succeed, the attacker must be able to determine which gate corresponds to the LSB bit of the 14^{th} round.

To effect a targeted attack, an attacker must first identify specific logic gates or wires in the circuit that implement the functionality that he wants to monitor and/or modify; for example, the gate or wire that corresponds to the privilege bit for the privilege escalation attack proposed in [19]. A malicious foundry can identify the functionality of every gate and wire in the circuit if it gets to fabricate the entire chip, i.e., if a conventional planar, 2D fabrication process is used. On the other hand, as we show in this paper, 3D integration significantly reduces the ability of an attacker in a malicious foundry to correctly identify gates or wires in the circuit that he wants to attack.

The specific 3D integration technology that we exploit in this work, since it is the only one that is currently in large volume commercial production [8], splits a design into two tiers. The bottom tier consists of digital logic gates and metal wires used to interconnect logic gates. The top tier, also referred to as an *interposer*, only consists of metal wires that provide additional connections between logic gates on the bottom tier.

The bottom tier — this tier is expensive to fabricate since it implements active transistor devices and passive metal — is sent to an external, untrusted foundry for fabrication. This is referred to as the untrusted tier. The top tier implements only passive metal and can be fabricated at lower cost in a trusted fabrication facility. We refer to this tier as the trusted tier.

Assume, for the sake of argument, that all interconnections between logic gates are implemented on the trusted tier, the attacker (who only has access to the untrusted tier) observes only a "sea" of disconnected digital logic gates. From the perspective of the attacker, gates of the same type, for example all NAND gates, are therefore indistinguishable from each other. (Assuming that the relative size or placement of gates reveals no information about interconnections between gates. This is addressed in Section 4.) Assume also that the attacker wants to attack a specific NAND gate in the circuit, and not just *any* NAND gate. The attacker now has two choices: (a) the attacker could randomly pick one NAND gate to attack from the set of indistinguishable NAND gates, and only succeed in the attack with a certain probability; or (b) the attacker could attack all indistinguishable NAND gates, primarily in cases where the attacker wants to monitor but not modify gates in the circuit, at the expense of a larger malicious circuit and thus, an increased likelihood of the attack being detected. In either instance, the attacker's ability to effect a malicious, targeted attack on the circuit is significantly hindered. We refer to this technique as *circuit obfuscation*.

In general, we define a *k-secure* gate as one that, from the attacker's perspective, cannot be distinguished from $k-1$ other gates in the circuit. Furthermore, a *k-secure* circuit is defined as one in which each gate is at least *k-secure*.

Contributions We make the following contributions:

- We propose a concrete way of leveraging 3D IC technology to secure digital ICs from an active attacker at the foundry. Whereas the use of 3D IC technology for security has been alluded to before, we are not aware of prior work like ours that discusses how it can be used meaningfully.
- We propose a formal notion of security in this context that we call *k*-security. We give a precise characterization of the underlying technical problems — computing k-security and deciding which wires to lift — and identify their computational complexity.

- We have devised a concrete approach to addressing the problem of lifting wires, which comprises a greedy heuristic to identify a candidate set of wires to be lifted, and the use of a constraint (SAT) solver to compute k-security.
- We have conducted a thorough empirical assessment of our approach on benchmark circuits, including a case-study of a DES circuit, that illustrates the inability of an attacker to effectively attack circuits secured using 3D IC based obfuscation.

2 Preliminaries and Related Work

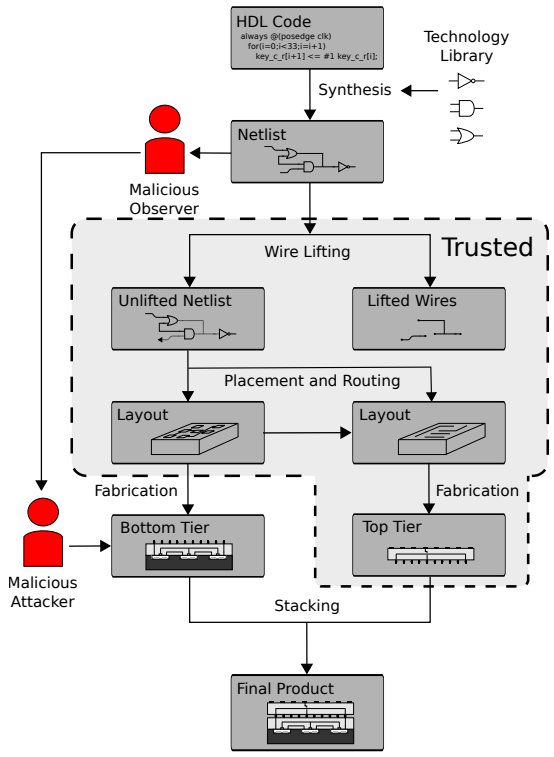


Figure 2: Secure 3D IC design and fabrication flow.

In this section, we overview the IC manufacturing process in the specific context of 3D integration, and discuss the attack model that we assume in this paper. We also discuss related work on hardware security including both attacks and countermeasures, and on the use 3D integration for enhancing the security of computer hardware.

2.1 3D IC Design and Fabrication

Digital ICs consist of a network of inter-connected digital logic gates. This network of gates is often referred to as a netlist. Digital logic gates are built using complementary metal-oxide-semiconductor (CMOS) transistors. In a conventional planar/2D IC, CMOS transistors, and by extension digital logic gates, lie in a single layer of silicon. In addition, there are several layers of metal wires used to inter-connect the gates.

3D integration enables the vertical stacking of two or more planar ICs. Each IC in the vertical stack is referred to as a tier. Vertical interconnects (TSVs) are provided to allow the transistors and metal wires in each tier to connect to each other.

The initial motivation for 3D integration came from the potential reduction in the average distance between logic gates — in a 3D IC, the third, vertical dimension can be used to achieve a tighter packing of logic gates [6]. However, a number of issues, including high power density, temperature and cost, have plagued high volume, commercial availability of logic-on-logic 3D ICs [13].

A more practical 3D IC technology that has been demonstrated in a commercial product (a Xilinx FPGA [8]) is shown in Figure 1. It consists of two tiers. The bottom tier contains both transistors/gates and metal wires, while the top tier, the interposer, contains only metal wires. The two tiers are interfaced using uniformly spaced metallic bond-points. TSVs make use of these bond-points to provide connections between wires in the top and bottom tiers. This technology has also been referred to as 2.5D integration [14]. In the rest of this paper, we use 3D instead of 2.5D since our techniques can easily be generalized to full 3D.

Since the bottom tier consists of CMOS transistors, it is fabricated at one of the few foundries worldwide with advanced lithographic capabilities at high cost. The top tier, i.e., the interposer, only contains passive metal and can be fabricated at significantly reduced cost [21].

Figure 2 shows a 3D IC design flow with appropriate modifications for security. The design flow begins with the design specified using a hardware description language (HDL), which is then synthesized to a netlist of gates. The types of gates allowed in the gate netlist are specified in a technology library.

In the **wire lifting** stage, the edges (or wires) in the netlist that are to be implemented on the top tier are selected. These are referred to as **lifted wires**. The rest of the netlist, implemented on the bottom tier, is referred to as the **unlifted netlist** and consists of unlifted gates and unlifted wires.

The unlifted gates are then placed on the surface of the bottom tier, i.e., the (x, y) co-ordinates for each gate are selected. Unlifted wires are routed using the bottom tier metal layers. Two bond-points are selected for every lifted wire; one each for the two gates that the wire connects. The gates are connected to the corresponding bond-points. Finally, lifted wires are routed between pairs of bond-points in the top tier using the top tier routing resources.

Finally, the two tiers are fabricated at separate foundries. The chips from the two foundries are vertically stacked to create the final 3D IC chip that is shipped to the vendor.

We now discuss the attack model that we address in this paper, in the context of the 3D design and fabrication flow outlined above.

2.2 Attack Model

The attack model that we address in this paper is that of a **malicious attacker** in the foundry. This attack model has been commonly used in the hardware security literature because of the serious threat it presents [18]. We further strengthen the attack by assuming a **malicious observer** in the design stage, working in collusion with the malicious attacker in the foundry.[1] The malicious observer has full knowledge of the circuit as it goes through the design process, but can not effect any changes. The malicious attacker in the foundry can, on the other hand, effect changes in the circuit layout before the chip is fabricated.

To defend against this attack, the following steps of the design and fabrication flow are assumed to be secure, i.e., executed by a trusted party: (a) the wire lifting, placement and routing steps in the design, and (b) the fabrication of the top tier (therefore also referred to as the trusted tier).

Discussion Three aspects of the attack and defense models deserve further mention. First, we note that the attack model described above subsumes a number of other practically feasible attack models. It is stronger than a malicious attacker in the foundry working by himself. It is also stronger than a malicious attacker in the foundry with partial design knowledge — for example, the attacker is likely to know the functionality and input/output behaviour of circuit he is attacking (an ALU or a DES encryption circuit, *etc.*). Providing the attacker with the precise circuit netlist can only strengthen the attack.

Second, the steps in the design and fabrication process that are assumed to be trusted are also relatively easy to perform in a secure manner, compared to the untrusted steps. Wire lifting and placement/routing (in the design stage) are performed using automated software tools, the former based on algorithms that we propose in this paper, and the latter using commercially available software from electronic design automation (EDA) vendors. In comparison, writing the HDL code is manually intensive, time-consuming and costly. Furthermore, only the top tier is fabricated in a trusted foundry. The top tier only consists of passive metal wires that are inexpensive compared to the active CMOS transistors and metal wires in the untrusted, bottom tier [21].

Finally, we assume that that all IC instances are manufactured before being sent out for stacking. If this were not the case, an attacker could intercept a stacked IC and reverse engineer the connections on the top tier. Armed with this knowledge, the attacker could then insert malicious hardware in future batches of the IC as they are being fabricated in the foundry.

[1] Note that 3D IC based circuit obfuscation cannot, and is not intended to, defend against malicious attackers in the design stage who can alter the HDL or circuit netlist.

2.3 Related Work

In this section, we discuss related work in the literature on hardware security and, specifically, the use of 3D ICs in this context. We also discuss the relationship of our work to database and graph anonymizing mechanisms.

Hardware Security Malicious circuits are expected to consist of two components, a trigger and the attack itself. The trigger for the attack can be based on data, for example when a specific cheat code appears at selected wires in the circuit [19], or on time, i.e., the trigger goes off after a certain period of time once the IC is shipped [33].

Once triggered, the malicious attack can either transmit or leak sensitive information on the chip, modify the circuit functionality or degrade the circuit performance. Tehranipoor and Koushanfar discuss a number of specific backdoors that fall within one of these categories [31].

Countermeasures against malicious attacks can be categorized in various ways. Design based countermeasures modify or add to the design of the circuit itself to provide greater security. These include N-variant IC design [4], data encryption for computational units [33] and adding run-time monitors to existing hardware [32]. Our work falls within this category. In contrast, testing based counter-measures use either pre-fabrication or post-fabrication testing and validation to detect, and in some cases, disable malicious circuits. A survey of these techniques can be found in [11].

Another way to categorize countermeasures is by their impact on the attack. Countermeasures to detect malicious circuits include IC fingerprinting [3] and unused circuit identification [17]. Some countermeasures can be used to disable malicious circuitry; for example, the power cycling based defense against timer triggers [33]. The proposed defense mechanism aims to deter attackers by hiding a part of the circuit and making it more difficult for the attacker to effect a successful attack.

3D Integration for Hardware Security Valamehr et al. [32] also exploit 3D integration capabilities to enhance the security of computer hardware, although in a manner orthogonal to ours. Their proposal involves adding a "control tier" on top of a regular IC to moni-

tor the activity of internal wires in the IC in a cost effective way. By monitoring internal wires on the chip, the control tier is able to detect potentially malicious activity and take appropriate recourse. Adding the monitors vertically on top of the IC to be protected reduces the power and performance cost of monitoring the IC. A similar technique was proposed by Bilzor [7].

Or technique exploits 3D integration in a different way, i.e., we use it to provide a malicious attacker in an IC foundry with an incomplete view of the circuit netlist, thus deterring the attack. Although the potential for this kind of defense mechanism has been alluded to before by Tezarron [1], ours is the first work, to our knowledge, to address this technique in any consequential way.

Hardware Obfuscation Hardware obfuscation techniques have been proposed to make circuits more difficult to reverse engineer. In particular, Roy et al. [26] augment a combinational circuit with key bits in such a way that the circuit only provides correct outputs when the key bits are set to pre-determined values. Rajendran et al. [24] further strengthen this defense mechanism by increasing the bar on the attacker to determine the secret key.

A difference between key-based circuit obfuscation mechanisms and circuit obfuscation via split manufacturing is that the notion of security in the former is conditioned on the computational capabilities of the attacker. In contrast, our notion of security is unconditional in that no matter the computational capabilities of the attacker, he cannot distinguish each gate from $k-1$ other gates. We note that these mechanisms are not necessarily mutually exclusive — it might be possible to leverage split manufacturing based circuit obfuscation to further strengthen key-based circuit obfuscation, or vice-versa.

Independent of this work, Rajendran et al. [25] have recently examined the security obtained from split manufacturing. However, the authors provide no well-founded notion of security for split manufacturing, as we do in this paper. The authors do not address the wire lifting problem at all, and implicitly assume that the circuit is partitioned using traditional min-cut partitioning heuristics. Finally, it is assumed that the attacker reconstructs the circuit by simply connecting the closest gates with disconnected inputs/outputs.

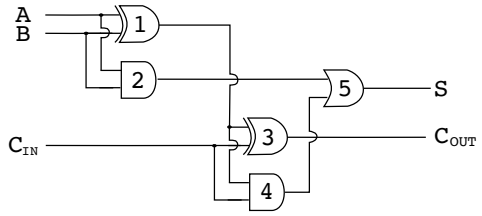
Anonymizing Databases and Social Networks Our work bears relationship to prior work on anonymizing databases and social network graphs, but also has significant differences. A database is k-anonymous if the information for each individual is indistinguishable from $k-1$ other individuals [30] in the database. The notion of k-anonymity for a social network is similar, except that instead of operating on relational data, it operates on a graph. Two individuals in a social network are indistinguishable if their local neighbourhoods are the same [34].

In our setting, the similarity of the local neighborhood of two gates is only a necessary but not sufficient condition for indistinguishability. This is because the attacker is assumed to have access to the original circuit netlist and an incomplete view of the same netlist, and must thus match *all* gates in the incomplete netlist to gates in the original netlist.

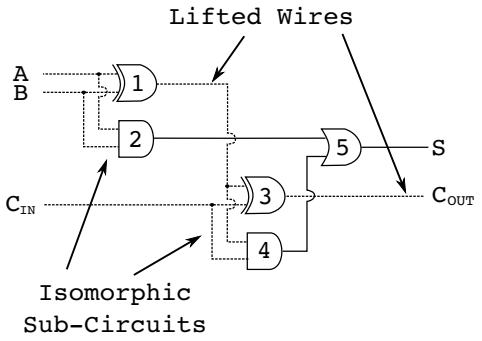
The circuit obfuscation problem also introduces a number of distinct practical issues. These include the additional information that might be conveyed by the circuit layout (for example, the physical proximity of gates), and the role of the number of gate types in the technology library.

3 Problem Formulation

In this section, we formulate the circuit obfuscation problem that we address in this paper as a problem in the context of directed graphs. We begin by discussing the example circuit for a full adder that we show in Figure 3.



(a) Original circuit netlist.



(b) Unlifted netlist.

Figure 3: Original and unlifted netlists corresponding to a full adder circuit. Grey wires in the unlifted netlist are lifted and are not observed by the attacker.

Example As we mention in Section 1, in the most powerful attack model we consider, an attacker is in possession of two pieces of information: the originally designed (complete) circuit netlist, and the layout of the circuit that is sent to the foundry for fabrication, which we call the unlifted netlist. The latter results from the

defender lifting wires from the former. Assume that the defender chooses to lift the wires $A \to \{1,2\}$, $B \to \{1,2\}$, $C_{IN} \to \{3,4\}$, $1 \to \{3,4\}$ and $3 \to C_{OUT}$.

Note that gates in the unlifted netlist in Figure 3(b) are labeled differently from those in the original circuit in Figure 3(a). This reflects the fact that the attacker obtains the original circuit netlist and the unlifted netlist in completely different formats. The original netlist is a set of gates and wires in HDL format. On the other hand, the unlifted netlist is reconstructed from the circuit layout, which is a set of shapes and their locations on the surface of the chip, as also discussed in Section 4.3. The labeling and ordering of objects in the circuit layout file is unrelated to that in the netlist of the original circuit. Although not required, the defender can perform an additional random re-labeling and re-ordering step before the layout of H is sent to the foundry.

Given these two pieces of information, the attacker now seeks a bijective mapping of gates in the unlifted netlist to gates in the complete circuit netlist. If the attacker is successful in obtaining the correct mapping, he can carry out a targeted attack on any gate (or gates) of his choosing. The security obtained from lifting wires in the context of this example can be explained as follows. From the attacker's perspective, either Gate u or Gate w in the unlifted netlist could correspond to Gate 1 in the original netlist. Thus the attacker's ability to carry out a targeted attack on Gate 1 is hindered. The same can be said for the attacker's ability to carry out a targeted attack on Gate 2, 3 or 4. However, note that the attacker can determine the identity of Gate 5 with certainty — it must correspond to Gate y since this is the only OR gate in the netlist. Thus, in this example, the lifting does not provide any security for Gate 5.

Informally, our notion of security is based on the existence of multiple isomorphisms (mappings) between gates in the unlifted netlist and the original netlist. In our example, there exist 4 distinct bijective mappings between the gates in the unlifted and original netlists. However, this notion of security may be seen as too permissive. It can be argued that given the fact that across all mappings, gate 5 is mapped uniquely, we have no security at all (i.e., security of 1). A more restrictive notion of security, one that we adopt in this paper, requires that for *each* gate in the original netlist, there exist at least k different gates in the unlifted netlist that map to it over all isomorphisms. This is intended to capture the intuition that the attacker is unable to uniquely identify even a single gate. We now formalize our notion of security.

3.1 Formulation as a Graph Problem

We now formulate our problem as a graph problem. A circuit can be perceived as a directed graph — gates are vertices, and wires are edges. The direction of an edge into or out of a vertex indicates whether it is an input or output wire to the gate that corresponds to the vertex. If G is a graph, we denote its set of vertices as $V[G]$, and its set of edges as $E[G]$. Each vertex in the graph is associated with a color that is used to distinguish types of gates (e.g, AND and OR) from one another. Consequently, a graph G is a 3-tuple, $\langle V, E, c\rangle$, where V is the set of vertices, E the set of edges and the function $c: V \to \mathbb{N}$ maps each vertex to a natural number that denotes its color. For example, the circuit in Figure 3 and its unlifted portion can be represented by the graphs in Figure 4.

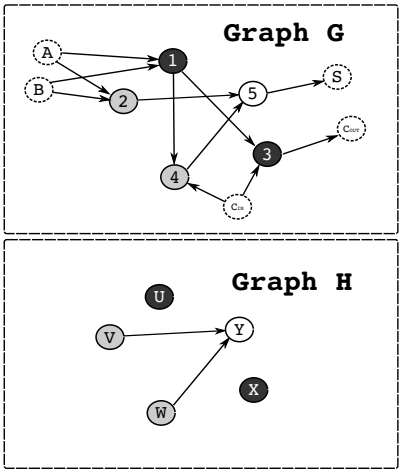


Figure 4: Full adder graphs: G is the full graph representation of the full adder circuit, H is the remaining graph after wires have been lifted.

A main challenge for the defender is to lift wires in a way that provides security. Our notion of security corresponds to a certain kind of subgraph isomorphism.

Definition 1 (Graph isomorphism)**.** *Given two graphs $G_1 = \langle V_1, E_1, c_1\rangle, G_2 = \langle V_2, E_2, c_2\rangle$, we say that G_1 is isomorphic to G_2 if there exists a bijective mapping $\phi: V_1 \to V_2$ such that $\langle u, v\rangle \in E_1$ if and only if $\langle \phi(u), \phi(v)\rangle \in E_2$ and $c_1(u) = c_2(\phi(u)), c_1(v) = c_2(\phi(v))$. That is, if we rename the vertices in G_1 according to ϕ, we get G_2. A specific such mapping ϕ is called an isomorphism.*

Definition 2 (Subgraph isomorphism)**.** *We say that $G_1 = \langle V_1, E_1, c_1\rangle$ is a subgraph of $G_2 = \langle V_2, E_2, c_2\rangle$ if $V_1 \subseteq V_2$, and $\langle u, v\rangle \in E_1$ only if $\langle u, v\rangle \in E_2$. We say that G is subgraph isomorphic to H if a subgraph of G is isomorphic to H. The corresponding mapping is called a subgraph isomorphism.*

For example, in Figure 4, a subgraph isomorphism, ϕ, is $\phi(1) = U, \phi(2) = V, \phi(3) = X, \phi(4) = W, \phi(5) = Y$.

Intuition Let G be the graph that represents the original circuit with all wires, and H the graph of the circuit

after wires have been lifted. Then, the attacker knows that G is subgraph isomorphic to H. What he seeks is the correct mapping of vertices in G to H (or vice versa). This is equivalent to him having reconstructed the circuit, and now, he can effect his malicious modifications to the circuit that corresponds to H.

From the defender's standpoint, therefore, what we seek intuitively is that there be several subgraph isomorphisms between G and H. As we mention in Section 1, this then gives the kind of security in a k-anonymity sense — the attacker cannot be sure which of the mappings is the correct one, and therefore is able to reconstruct the circuit with probability $1/k$ only. As we mention there and discuss in more detail in the related work Section, though our notion of security has similarities to k-anonymity, there are important differences, and we call it k-security instead.

k-security We now specify our notion of security. We do this in three stages. (1) We first define a problem that captures our intuition of a gate being indistinguishable from another gate. We do this by requiring the existence of a particular kind of subisomorphic mapping between graphs that represent circuits. (2) We then define the notion of a k-secure gate. Such a gate is indistinguishable from at least $k-1$ other gates in the circuit. (3) Finally, we define the notion of k-security, which is security across all gates in the circuit. This definition requires simply that every gate in the circuit is k-secure.

In the following definition, we characterize the problem GATE-SUBISO, which captures (1) above — a notion of what it means for a gate to be indistinguishable from another.

Definition 3 (GATE-SUBISO)**.** *Given as input $\langle G, E', u, v\rangle$, where G is a DAG, $E' \subseteq E[G]$, and two distinct vertices $u, v \in V[G]$, let H be the graph we get by removing the edges that are in E' from G. Then, GATE-SUBISO is the problem of determining whether there exists a mapping $\phi : V[G] \to V[H]$ that is a subgraph isomorphism from G to H such that $\phi(u) = v$.*

The above definition is a special case of the well-known subgraph isomorphism problem [16]. In the subgraph isomorphism problem, we take as input two graphs A, B, and ask whether B is subgraph isomorphic to A. In GATE-SUBISO, both the graphs G, H are restricted to be DAGs, and H is a specific subgraph of G — one with some edges removed from G. Of course, we know that H is subgraph isomorphic to G, with the identity mapping from a vertex to itself serving as evidence (a certificate). However, in the GATE-SUBISO problem, we require the existence of a subgraph isomorphism that is different from the identity mapping, and furthermore, require that the vertex u be mapped under that subgraph isomorphism to a specific vertex v.

The intuition behind GATE-SUBISO is the following. G is the graph that corresponds to the original circuit, and H is the graph that corresponds to the circuit after wires are lifted. The above definition for GATE-SUBISO asks whether there exists a mapping under which the vertex u in the original circuit is indistinguishable from v in the unlifted circuit. That is, given that $u \neq v$, an attacker does not know whether u in G corresponds to u or v in H.

Based on GATE-SUBISO above, we now define the notion of a k-secure gate. It captures the intuition that the gate is indistinguishable from at least $k-1$ other gates.

Definition 4 (k-secure gate)**.** *Given a DAG, G, a vertex u in it, and a subgraph H of G constructed from G by removing some edges, $E' \subseteq E[G]$ only. We say that u is k-secure if there exist k distinct vertices $v_1, \ldots, v_k$ in G (and therefore in H), and mappings $\phi_1, \ldots, \phi_k$ from $V[G]$ to $V[H]$ such that every ϕ_i is a subgraph isomorphism from G to H, and for all $i \in [1,k]$, $\phi_i(u) = v_i$.*

The above definition expresses that u is indistinguishable from each of the v_i's. Of course, one of the v_i's may be u itself. Therefore, every gate is 1-secure, and if a gate is not 2-secure, then that gate is uniquely identifiable, for this particular choice of E'. The maximum that k can be is $|V[G]|$, the number of a vertices in G.

Given the above definition for a k-secure gate, it is now straightforward to extend it to the entire graph (circuit). We do this with the following definition.

Definition 5 (k-security)**.** *Given a DAG G, and a DAG H that we get from G by removing the edges from a set $E' \subseteq E[G]$. We say that $\langle G, E'\rangle$ is k-secure if every vertex in G is k-secure.*

The above definition is a natural extension of the notion of a k-secure gate, to every gate in the circuit. What it requires for k-security is that every vertex in the corresponding graph is indistinguishable from at least k vertices. We point out that some gates may be more than k-secure; k-security is a minimum across all gates. As the maximum k for any gate is $|V[G]|$, a graph can be, at best, $|V[G]|$-secure. Every graph is 1-secure, which is the minimum.

We denote as $\sigma(G, E')$ the maximum k-security we are able to achieve with G, E'. In Figure 4, for example, we know that σ evaluates to 1, because the node 5 can be mapped to itself only. The nodes 1, 2, 3 and 4, however, are 2-secure gates. The reason is that each can be mapped either to itself, or to another node.

Computational complexity We now consider the computational complexity of determining the maximum k-security, σ. We consider a corresponding decision problem, k-SECURITY-DEC, which is the following. We are given as input $\langle G, E', k\rangle$ where G is a DAG, $E' \subseteq E[G]$

is a set of edges in G, and $k \in [1, |V[G]|]$. The problem is to determine whether lifting the edges in E' results in k-security.

We point out that if we have an oracle that decides k-SECURITY-DEC, then we can compute the maximum k-security we can get by lifting E' from G using binary search on k. That is, the problem of computing σ is easy if deciding k-SECURITY-DEC is easy.

Theorem 1. *k-SECURITY-DEC $\in$ **NP**-complete under polynomial-time Turing reductions.*

To prove the above theorem, we need to show that k-SECURITY-DEC is in **NP**, and that it is **NP**-hard. For the former, we need to present an efficiently (polynomial-sized) certificate that can be verified efficiently. Such a certificate is k mappings each of which is a subgraph isomorphism, for each vertex $u \in V[G]$. Each such mapping can be encoded with size $O(|V[G]|)$, and there are at most $k|V[G]|$ such mappings, and therefore the certificate is efficiently-sized. The verification algorithm simply checks that each mapping is indeed a subgraph isomorphism, and that u is mapped to a distinct vertex in each of the k mappings that corresponds to it. This can be done in time $O(|V[G]|^3)$.

We show that k-SECURITY-DEC is **NP**-hard under polynomial-time Turing reductions in the Appendix. (Henceforth, we drop the qualification "under polynomial-time Turing reductions," and simply say **NP**-complete and **NP**-hard.) Indeed, our proof demonstrates that deciding even 2-security is **NP**-hard. The knowledge that k-SECURITY-DEC is **NP**-complete immediately suggests techniques for approaches for solving k-SECURITY-DEC, and thereby computing k-security. We discuss this further in the next section.

Choosing E' Lifting edges E' from G incurs a cost $c(G,E')$. A simple cost metric, one that we adopt in this paper is $c(G,E') = |E'|$, i.e., the cost is proportional to the number of lifted edges. Given the cost of lifting edges, the defender's goal is to determine E', the set of edges that should be lifted, such that $\sigma(G,E') \geq k$ and $c(G,E')$ is minimized.

We observe that from the standpoint of computational complexity, the problem of determining E' given G,k, where G is the graph and E' is the set of edges to be lifted so we get k-security, is no harder than k-SECURITY-DEC. That is, that problem is also in **NP**.

To prove this, we need to show that there exists an efficiently sized certificate that can be verified efficiently. Such a certificate is E', and k subgraph isomorphisms for every vertex. The latter component of the certificate is the same as the one we used in our proof above for k-SECURITY-DEC's membership in **NP**. The verification algorithm, in addition to doing what the verification algorithm for k-SECURITY-DEC above does, also checks that E' is indeed a subset of G's edges.

We note that the k-security from lifting all the edges in G is no worse than lifting any other set of edges, and the k-security from lifting no edges in G is no better than lifting any other set of edges. More generally, given any n_1, n_2 such that $|E[G]| \geq n_1 > n_2$, we know that for every G, there exists a set of edges of size n_1 that if lifted, provides at least as much security as every set of edges of size n_2. That is, there is a natural trade-off between the number of edges we lift, i.e., cost, and the security we achieve. In Section 4, we outline an approach to determine the cost-security trade-off using a greedy wire lifting procedure.

3.2 Discussion

Given our notion of k-security, a natural question to ask is whether there are stronger or different attack models for which k-security would be inadequate. We discuss this in the context of two attack models that differ from the one assumed. Finally, we also discuss a related question — that of the computational capabilities of the attacker.

General targeted attack models The notion of k-security is premised on an attack model in which the attacker needs to precisely identify one or more gates in the unlifted netlist, for example, the privilege escalation bit in a microprocessor [19] or the LSB of the 14^{th} round in a DES implementation [9]. However, one can imagine a scenario in which the attack would be successful if the attacker correctly identifies any one of n gates. For example, there could be multiple privilege escalation bits in the microprocessor implementation.

More concretely, in the example in Figure 3, assume that the attacker wants to change the circuit functionality by inverting the output of Gate 2. The same objective can be accomplished by inverting the output of Gate 4. However, as we observe before, Gate v in the unlifted netlist must correspond to either Gate 2 or Gate 4. Thus, although this gate is 2-secure, the attack would be successful with probability 1.

Although our notion of security does not directly address the alternate attack model described above, it can be easily modified to do so. Say that the defender is aware that Gate v and Gate x are each equally vulnerable to the same kind of attack. Then, the defender can insist that Gate v is k-secure if and only if it is indistinguishable from $k-1$ other gates excluding Gate x. Such information that the defender may have about the relative vulnerability of gates can be built into the notion of k-security.

Access to lifting procedure Our attack model strengthens the attacker with access to the original cir-

cuit netlist, G, along with the unlifted netlist H. Since the attacker has access to G, it is reasonable to ask if an even stronger attacker with access to G *and* the procedure used to lift wires would compromise security. It would not.

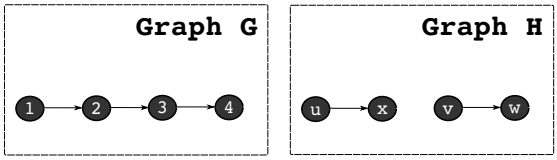


Figure 5: Example illustrating that the unlifted netlist H is 2-secure even if the attacker knows that edge $2 \rightarrow 3$ was lifted from original netlist G.

In fact, even if there is a deterministic choice of edges that must be lifted to provide a certain security level, knowledge of which edges are lifted does not compromise security, as long as G and H are differently labeled. We illustrate this with an example in Figure 5, where wire $2 \rightarrow 3$ must be lifted to provide 2-security. This knowledge does not compromise the security obtained from lifting. When there is choice, i.e., lifting two or more edges provides the same security, the choice is made uniformly at random. This is discussed in Section 4.

Computational capabilities of the attacker Our notion of k-security is not predicated on the computational capabilities of the attacker. In fact, we assume that the attacker is able to identify (all) subgraph isomorphisms from H to G. Nonetheless, given that the attacker's goal might be to identify a single gate in the netlist, it is natural to ask why (and whether) the attacker needs determine a mapping for each gate in H.

In particular, the attacker can identify all gates in H with the same type and connectivity, i.e., number and type of gates it connects to, as the one he is interested in attacking. Prior work on k-anonymity for social network graphs assumes this kind of attack strategy. From the perspective of the attacker, this strategy is sub-optimal. This is because, for any gate in G that the attacker wants to target, this strategy will provide at least as many candidate mappings in H as the strategy in which the attacker enumerates all subgraph isomorphisms.

4 Approach

Having considered the computational complexity of the problem that underlies our work in the previous section, in this section, we propose a concrete approach for it. As our discussions in the prior section reveal, there are two parts to the solution: (a) computing the maximum k-security for $\langle G, E' \rangle$, given the graph G that represents the complete circuit, and, (b) choosing the set E'.

We propose an approach for each in this section. For the problem of computing security, we employ constraint-solving. We discuss this in Section 4.1. For the problem of choosing E', we propose a greedy heuristic. We discuss that in Section 4.2. We conclude this section with Section 4.3 with some practical considerations, specifically, scalability and layout-anonymization.

4.1 Computing Security

As shown in Section 3, the problem of determining the security level of circuit G, given the unlifted netlist H is **NP**-complete. Given the relationship of the problem to subgraph isomorphism, a natural approach to solving this problem would be to use graph (sub)isomorphism algorithms proposed in literature — of these, the VF2 algorithm [12] has been empirically shown to be the most promising [15]. However, in our experience, VF2 does not scale for circuits with > 50 gates (more on scalability in Section 4.3).

Instead, motivated by the recent advances in the speed and efficiency of SAT solvers, we reduce the sub-isomorphism problem to a SAT instance and use an off-the-shelf SAT solver to decide the instance.

Reduction to SAT Given graphs G and H, we define a bijective mapping ϕ from the vertex set of H to the vertex set of G as follows: Boolean variable ϕ_{ij} is true if and only if vertex $q_i \in H$ maps to a vertex $r_j \in G$. Here $V[G] = \{r_1, r_2, \ldots, r_{|V[G]|}\}$ and $V[H] = \{q_1, q_2, \ldots, q_{|V[H]|}\}$

We now construct a Boolean formula that is true if and only if graphs G and H are sub-isomorphic for the mapping ϕ. We will construct the formula in parts.

First, we ensure that each vertex in G maps to only one vertex in H:

$$F_1 = \prod_{i}^{|V[H]|} \sum_{j}^{|V[G]|} \left(\phi_{i,j} \prod_{k \neq i}^{|V[G]|} \neg \phi_{i,k} \right)$$

and vice-versa:

$$F_2 = \prod_{j}^{|V[G]|} \sum_{i}^{|V[H]|} \left(\phi_{i,j} \prod_{k \neq i}^{|V[H]|} \neg \phi_{k,j} \right)$$

Finally we need to ensure that each edge in H maps to an edge in G. Let $E[H] = \{e_1, e_2, \ldots, e_{|E[H]|}\}$ and $E[G] = \{f_1, f_2, \ldots, f_{|E[G]|}\}$. Furthermore, let $e_k = \langle q_{src(e_k)}, q_{dest(e_k)} \rangle \in E[H]$ and $f_k = \langle r_{src(f_k)}, r_{dest(f_k)} \rangle \in E[G]$. This condition can be expressed as follows:

$$F_3 = \prod_{k}^{|E[H]|} \sum_{l}^{|E[G]|} \phi_{src(e_k),src(f_l)} \wedge \phi_{dest(e_k),dest(f_l)}$$

The formula F that is input to the SAT solver is then expressed as a conjunction of the three formulae above: $F = F_1 \wedge F_2 \wedge F_3$. The formula F has $O(|V[H]||V[G]|)$ variables and $O(|E[H]||E[G]|)$ clauses.

4.2 Wire Lifting Procedure

To determine a candidate set of edges, E', to lift, we employ a greedy heuristic. Our heuristic is shown as Algorithm 1.

```
1  E' ← E[G]
2  while |E'| > 0 do
3      s ← 0
4      foreach e ∈ E' do
5          E' ← E' − {e}
6          if σ(G,E') > s then
7              s ← σ(G,E')
8              e_b ← e
9          E' ← E' ∪ {e}
10     if s < k then return E'
11     E' ← E' − {e_b}
12 return E'
```

Algorithm 1: lift_wires(G, k)

In our heuristic, we begin with the best security we can achieve. This occurs when we lift every edge in $E[G]$; that is, we set E' to $E[G]$ at the start in Line 1. We then progressively try to remove edges from E', in random order. We do this if not lifting a particular edge e still gives us sufficient security.

That is, we iterate while we still have candidate edges to add back (Line 2). If we do, we identify the "best" edge that we can add back, i.e., the one that gives us the greatest security level if removed from E'. If even the best edge cannot be removed from E', then we are done (Line 10).

The heuristic does not necessarily yield an optimal set of edges. The reason is that we may greedily remove an edge e_1 from E' in an iteration of the above algorithm. And in later iterations, we may be unable to remove edges e_2 and e_3. Whereas if we had left e_1 in E', we may have been able to remove both e_2 and e_3. Note that removing as many edges from E' is good, because our cost is monotonic in the size of E' (set of edges being lifted).

4.3 Practical Considerations

From a graph-theoretic perspective, the wire lifting procedure outlined provides a set of wires to lift that guarantees a certain security level. However, two practical considerations merit further mention — the scalability of the proposed techniques to "large" circuits, and the security implication of the attacker having access to the layout of H, as opposed to just the netlist.

Scalability Although the SAT based technique for computing security scales better than the VF2 algorithm, we empirically observe that it times out for circuits with > 1000 gates. To address this issue, we propose a circuit partitioning approach that scales our technique to larger circuits of practical interest. We note that circuit partitioning is, in fact, a commonly used technique to address the scalability issue for a large number of automated circuit design problems.

Algorithm 2 is a simplified description of the partitioning based wire lifting procedure. The function $partition(G)$ recursively partitions the vertex set of the graph into P mutually exclusive subsets and returns subgraphs $\{G_1, G_2, \ldots, G_P\}$ of size such that they can be tractably solved by the SAT based greedy wire lifting procedure. The final set of lifted wires includes the union of all wires that cross partitions, and those returned by P calls to Algorithm 1. We have used this technique to lift wires from circuits with as many as 35000 gates (see Section 5).

```
1  {G_1, G_2, ..., G_P} ← partition(G)
2  E_R ← E − ⋃_{i∈[1,P]} E_i
3  for i ∈ [1,P] do
4      E_R ← E_R ⋃ lift_wires(G_i, s_req)
5  return E_R
```

Algorithm 2: lift_wires_big(G, s_{req})

Layout anonymization We have, so far, assumed that the unlifted circuit H is a netlist corresponding to the unlifted gates and wires. However, in practice, the attacker observes a layout corresponding to H, from which he reconstructs the netlist of H. We therefore need to ensure that the layout does not reveal any other information to the attacker besides the reconstructed netlist.

Existing commercial layout tools place gates on the chip surface so as to minimize the average distance, typically measured as the Manhattan distance, between all connected gates in the circuit netlist. Thus, if the complete circuit G is used to place gates, the physical proximity of gates will reveal some information about lifted wires — gates that are closer in the bottom tier are more likely to be connected in the top tier. The attacker can use this information to his advantage.

Instead of using the netlist G to place gates, we instead use the netlist H. Since this netlist does not contain *any* lifted wires, these wires do not have any impact on the resulting placement. Conversely, we expect the physical proximity of gates to reveal no information about hidden

wires in the top tier. In Section 5, we empirically validate this fact. However, anonymizing the layout with respect to the hidden wires does result in increased wire-length between gates, which has an impact on circuit performance. This impact is also quantified in Section 5.

5 Results

We conduct our experimental study using two exemplar benchmarks, the c432 circuit from the ISCAS-85 benchmark suite [10] (a 27-channel bus interrupt controller) with $\approx$ 200 gates, and a larger DES encryption circuit with $\approx$ 35000 gates. We use the c432 circuit to investigate security-cost trade-offs obtained from the proposed techniques and use the larger DES circuit for a case study.

All experimental results are obtained using an IBM 0.13μ technology. For 3D integration, bond points are assumed to be spaced at a pitch of $4\mu m$, allowing for one bond-point per $16\mu m^2$. This is consistent with the design rules specified in the Tezzaron $0.13\mu m$ technology kit.

Circuit synthesis was performed using the Berkeley SIS tool [27]. Placement and routing is performed using Cadence Encounter. Finally, we used miniSAT as our SAT solver [29].

5.1 Security-Cost Trade-offs

Figure 6 graphs the security level for the c432 circuit as a function of $E[H]$, the number of unlifted wires in the untrusted tier. $E[H] = 0$ corresponds to a scenario in which *all* wires are lifted, while $E[H] = E[G]$ corresponds to a case in which all wires are in the untrusted tier.

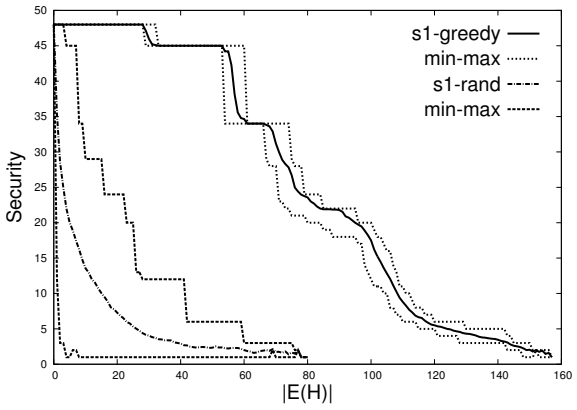


Figure 6: Maximum, average and minimum security levels for the c432 circuit using the proposed greedy wire lifting procedure and random wire lifting.

Proposed Vs. Random Wire Lifting Figure 6 compares the proposed greedy wire lifting technique with a baseline technique in which wires are lifted at random. In both cases, we show the maximum, average and minimum security achieved by these techniques over all runs.

Observe that greedy wire lifting provides significantly greater security compared to random wire lifting. With 80 unlifted wires, the greedy solution results in a 23-secure circuit, while *all* random trials resulted in 1-secure (equivalently, completely insecure) circuits.

Number of Lifted Edges vs. Security Figure 6 reveals that, for c432, at least 145 of the 303 ($\approx$ 47%) wires must be lifted to get any meaningful degree of security. If any fewer wires are lifted, circuit obfuscation provides no security at all. However, once more than this minimum number of wires is lifted, the security offered increases quite rapidly.

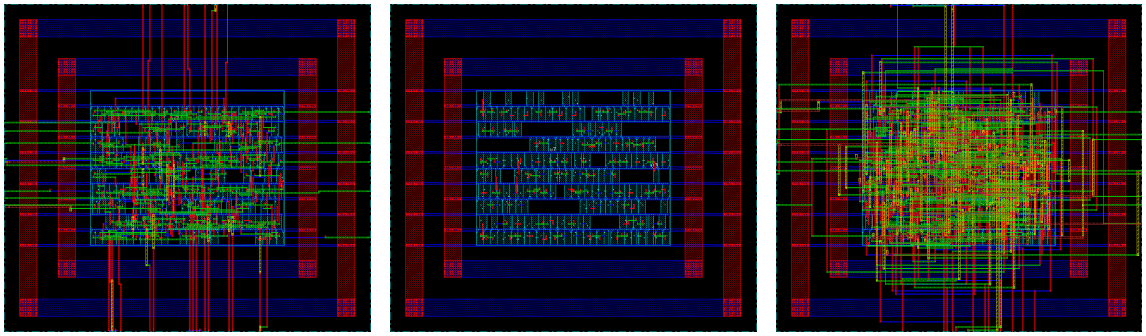
Another observation that merits mention are the plateaus in security level, for example between $E[H] = 30$ and $E[H] = 55$. In other words, in some cases, wires can be retained in the untrusted tier without any degradation in security.

Impact of Layout Anonymization Figure 7 shows three layouts for the c432 circuit. The far left corresponds to the original 1-secure c432 circuit without any wire lifting. The other two layouts correspond to the top and bottom tiers of an 8-secure version of c432 with $\approx 66\%$ lifted wires. Of particular interest is the wire routing in the trusted top tier — because the placement of the corresponding gates in the untrusted bottom tier have been anonymized, the lifted wires are routed seemingly randomly. This is in stark contrast to the wire routing in the original circuit that is far more structured.

Figure 8 shows the histogram of wire lengths for the three layouts shown in Figure 7. Note that, in the original 1-secure circuit, a large majority of wires are short; in other words, connected gates are placed closer together. Wire lengths on the bottom untrusted tier of the 8-secure circuit also skew towards shorter values — however, these wires are already observable to the attacker and he gains no additional information from their lengths. On the other hand, the wire length distribution of the top tier is more evenly spread out. This reflects that fact that the physical proximity of gates in the bottom tier reveals very little information about the lifted wires.

A Chi Square test was performed to determine if the distribution of wirelengths in the top tier is different from one that would be obtained from a random placement of gates. The test does not provide any evidence to reject the null hypothesis ($N = 11$, $\chi^2 = 0.204$ and $p = 0.999$), i.e., it does not reveal any significant difference between the two distributions.

Area, Delay and Power Cost Area, delay (inversely proportional to clock frequency) and power consumption are important metrics of circuit performance. 3D integra-



(a) Original Circuit (b) Bottom Tier of 8-Secure Circuit (c) Top Tier of 8-Secure Circuit

Figure 7: Layout of c432 without any lifting (left), and the bottom (middle) and top (right) tiers of an 8-secure version of c432. Green and red lines correspond to metal wires.

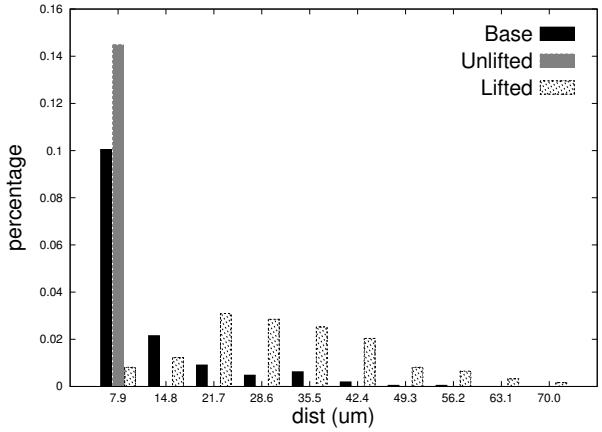


Figure 8: Comparison of the c432 circuit wire lengths the original 1-secure circuit and the bottom and top tiers of the 8-secure circuit.

tion based circuit obfuscation introduces overheads on all three metrics.

The area of a 3D circuit is determined by the larger of two areas: the area consumed by the standard cells in the bottom tier, and the area consumed by the bond-points required to lift wires to the top tier. The bond-point density is limited by technology (1 bond-point per $16\mu m^2$ in our case) and therefore more lifted wires correspond to increased area.

Delay and power are strong functions of wire length, as increased wire length results in increased wire capacitance and resistance. Layout anonymization results in increased wire length as we have observed before.

Table 1 shows the area, power and delay for the c432 circuit for different security levels. Compared to the original circuit, the 8-secure circuit has $1.6\times$ the power consumption, $1.8\times$ delay, and about $3\times$ the area.

Choice of Technology Library The technology library determines the type of gates that are allowed in the circuit netlist. Diverse technology libraries with many different gate types allow for more optimization, but also hurt security. Figure 9 shows the security levels achievable for c432 for five different technology libraries with between three and seven gates.

5.2 Case Study: DES Circuit

We use the DES encryption benchmark circuit to demonstrate that applicability of our techniques, including circuit partitioning based wire lifting, to larger circuits. The DES circuit takes as input a fixed-length string of plaintext and transforms the string into cipher text using 16 rounds of obfuscation, as shown in the block-level circuit diagram in Figure 10.

The original, 1-secure implementation of DES that we synthesized has $\approx$ 35000 logic gates, which results in an intractable SAT instance. However, using recursive circuit partitioning, we are able to lift wires to obtain a 64-secure implementation. We note that a security level of 16 is obtained in the first few rounds of partitioning by

Table 1: Power, delay, wire length and area analysis for different levels of security on the c432 circuit. 1^* is the base circuit with no wires lifted and 48^* has all of the wires lifted.

Security	Power Ratio	Delay Ratio	Total Wire Length (μm)	Total Area (μm^2)
1^*	1.00	1.00	2739	1621
2	1.54	1.73	6574	4336
4	1.55	1.76	7050	4416
8	1.61	1.82	8084	4976
16	1.62	1.86	8161	5248
24	1.71	1.98	9476	6048
32	1.73	1.99	9836	6368
48^*	1.92	2.14	13058	8144

Table 2: Technology libraries used for the experiment in Figure 9. lib-x corresponds to a library with x different gate types.

Library	max(S_1)	$\|V(G)\|$	$\|E(G)\|$	Gates
lib-3	48	209	303	inv, nor, nand
lib-4	24	181	271	+nand_3
lib-5	13	169	259	+nor_3
lib-6	7	165	252	+nand_4
lib-7	4	159	246	+nor_4

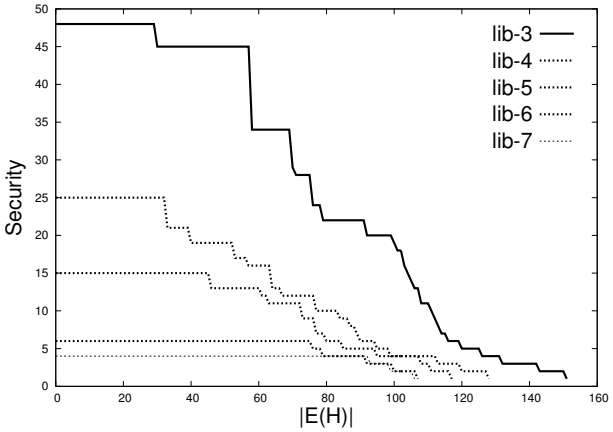


Figure 9: Obtainable security levels for the c432 circuit with different technology libraries.

removing only 13% of the wires, i.e., all wires that lie between successive DES rounds. This is because the circuit description of each DES round is identical — thus, once the wires between the rounds have been removed, each round can be confused for any other round. The final 64-secure implementation has only 30% of the wires unlifted, and consumes 2.38× the area of the original 1-secure circuit.

Attack Scenario Boneh et al. [9] have shown that specific bits in a DES implementation are particularly susceptible to fault attacks. For example, if the attacker is able to insert an attack such that the LSB output of the 14^{th} round is stuck at logic zero, the secret key can be recovered using as few as two messages.

Figure 11 shows how such an attack might be effected using a trigger (we do not address here how this trigger may be activated) and three additional gates in an insecure (or 1-secure) circuit. When the trigger is set, the output is set to zero, but is equal to the correct value when the trigger is at logic zero.

Now, assume that wire lifting is performed to make the circuit 64-secure. Given the set of lifted wires, we note that the LSB of the 14^{th} round is, in fact, 256-secure, i.e., there are 255 other gates in the circuit that are indistinguishable from the LSB of the 14^{th} round.

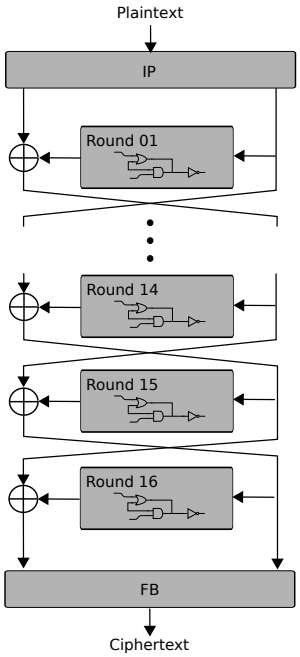


Figure 10: Block diagram of the DES encryption circuit.

The attacker now has two choices. he can either attack one of the 256 options, and only succeed with probability $\frac{1}{256}$, or he can choose to carry out a multiplexed attack on all 256 gates. This is shown in Figure 11. In this attack, the trigger transmits a sequence of 8-bits that identify which of the 256 signals the attacker wants to attack. These 8-bits feed an 8:256 demultiplexer that generates individual triggers for each of the 256 signals that are indistinguishable.

The attacker can now iteratively insert attacks in each gate one at a time and conceivably determine which iteration actually corresponds to the LSB of the 14^{th} round. However, in doing so, the attacker incurs two costs: (i) the modified attack circuit now requires 1280 gates instead of just 3, a 420× overhead; (ii) the attacker would require, in the worst case 255× more messages to recover the key.

5.3 Discussion

We have so far illustrated the quantitative trade-off between cost and security using benchmark circuits. We now discuss this trade-off qualitatively. In particular, we address aspects relating to both the security that 3D IC based split manufacturing can provide and the cost that it incurs in doing so.

From a security standpoint, we note that our notion of k-security is conservative. This is for two reasons. First, we have assumed a strong attack model in which the attacker has access to the original circuit netlist. In prac-

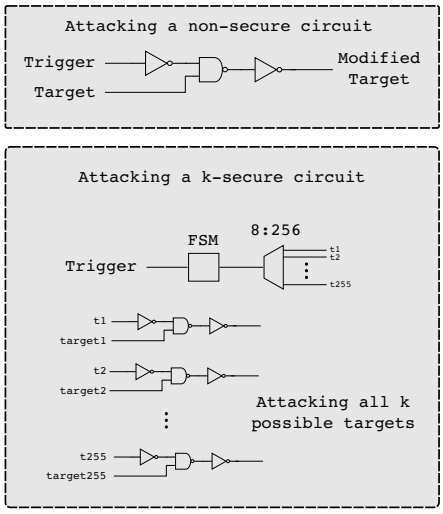


Figure 11: Attack scenarios of 1- and k-secure circuits.

tice, the attacker might only have access to the Boolean functionality of the circuit under attack, but not its gate level implementation. Second, in realistic attack scenarios, the attacker might need to identify more than one gate in the netlist. In both settings k-security serves as a lower bound on the security obtained from 3D IC based split manufacturing.

Furthermore, hardware attacks that are inserted in the foundry are different from other attack scenarios in that they are single shot, and require more effort, risk and expense to carry out. Thus, even relatively low values of k are likely to act as a significant deterrent for the attacker. If the attacker picks one gate to attack at random from the candidate set, he is only successful with probability $\frac{1}{k}$ and receives a payoff which is greater than his cost. However, with probability $\frac{k-1}{k}$, the attacker incurs a (significant) cost and receives no payoff. With $k = 100$ for example, the attacker's payoff must be $> 99\times$ his cost for him to break even (on average). Alternatively, the attacker could try attacking all 100 gates that are candidate mappings for his desired target (as shown in Figure 11), but this would incur a significantly increased risk of detection during post-fabrication testing.

From a cost standpoint, our empirical evaluations suggest a $1.5\times - 2\times$ overhead in area, performance (performance is proportional to circuit delay) and power consumption, which is the price we pay for security. Although there is relatively little work in this area, these overheads compare well to those of competing solutions such as field programmable gate arrays (FPGAs). In an FPGA, the desired circuit netlist is programmed on the FPGA *after* fabrication, so an attacker in a foundry receives no information about the circuit the designer wants to implement. However, benchmark studies have shown that FPGAs are $20\times$, $12\times$ and $4\times$ worse than custom digital ICs in terms of area, power and performance, respectively [20]. In addition, the FPGA itself could be attacked during fabrication in a way that allows an attacker in the field (after fabrication) to recover the circuit that has been programmed on it.

Finally, we note that the proposed technique can be selectively applied to only small, security critical parts of the design. Thus the area, performance and power overheads of split manufacturing would be amortized over the parts of the design that are conventionally implemented. It might also be possible to use split manufacturing in conjunction with other security techniques proposed in the literature such as key-based obfuscation [26, 24]. Key-based obfuscation is only conditionally secure, conditioned on the attacker's computational capabilities. We believe that split manufacturing can be used to further strengthen key-based obfuscation and make it unconditionally secure, although we leave this investigation as future work.

6 Conclusion

In this paper, we have proposed the use of 3D integration circuit technology to enhance the security of digital ICs via circuit obfuscation. The specific 3D technology we exploit allows gates and wires on the bottom tier, and only metal wires on the top. By implementing a subset of wires on the top tier, which is manufactured in a trusted fabrication facility, we obfuscate the identity of gates in the bottom tier, thus deterring malicious attackers.

We introduce a formal notion of security for 3D integration based circuit obfuscation and characterize the complexity of computing security under this notion. We propose practical approaches to determining the security level given a subset of lifted wires, and of identifying a subset of wires to lift to achieve a desired security level. Our experimental results on the c432 and DES benchmark circuits allow us to quantify the power, area and delay costs to achieve different security levels. In addition, we show, using a DES circuit case study, that 3D IC based circuit obfuscation can significantly reduce the ability of an attacker to carry out an effective attack.

Acknowledgements

We thank our shepherd, Cynthia Sturton, and the anonymous reviewers for their feedback and comments. We thank also Vijay Ganesh and Supreeth Achar for their inputs at the initial stages of this research. The work was supported by funding from the NSERC Discovery and Strategic grant programs.

References

[1] 3D-ICs and Integrated Circuit Security. Tech. rep., Tezzarron Semiconductors, 2008.

[2] ADEE, S. The hunt for the kill switch. *IEEE Spectrum 45*, 5 (may 2008), 34 –39.

[3] AGRAWAL, D., BAKTIR, S., KARAKOYUNLU, D., ROHATGI, P., AND SUNAR, B. Trojan detection using IC fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy* (2007), IEEE, pp. 296–310.

[4] ALKABANI, Y., AND KOUSHANFAR, F. N-variant IC design: Methodology and applications. In *Proceedings of the 45th annual Design Automation Conference* (2008), ACM, pp. 546–551.

[5] ARORA, S., AND BARAK, B. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[6] BANERJEE, K., SOURI, S. J., KAPUR, P., AND SARASWAT, K. C. 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proceedings of the IEEE 89*, 5 (2001), 602–633.

[7] BILZOR, M. 3D execution monitor (3D-EM): Using 3D circuits to detect hardware malicious inclusions in general purpose processors. In *Proceedings of the 6th International Conference on Information Warfare and Security* (2011), Academic Conferences Limited, p. 288.

[8] BOLSENS, I. 2.5D ICs: Just a stepping stone or a long term alternative to 3D? In *Keynote Talk at 3-D Architectures for Semiconductor Integration and Packaging Conference* (2011).

[9] BONEH, D., DEMILLO, R., AND LIPTON, R. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology – EUROCRYPT* (1997), Springer, pp. 37–51.

[10] BRGLEZ, F. Neutral netlist of ten combinational benchmark circuits and a target translator in fortran. In *Special session on ATPG and fault simulation, Proc. IEEE International Symposium on Circuits and Systems, June 1985* (1985), pp. 663–698.

[11] CHAKRABORTY, R. S., NARASIMHAN, S., AND BHUNIA, S. Hardware trojan: Threats and emerging solutions. In *Proceedings of the IEEE International Workshop on High Level Design Validation and Test (HLDVT)* (2009), IEEE, pp. 166–171.

[12] CORDELLA, L. P., FOGGIA, P., SANSONE, C., AND VENTO, M. Performance evaluation of the vf graph matching algorithm. In *Proceedings of the International Conference on Image Analysis and Processing* (1999), IEEE, pp. 1172–1177.

[13] DAVIS, W. R., WILSON, J., MICK, S., XU, J., HUA, H., MINEO, C., SULE, A. M., STEER, M., AND FRANZON, P. D. Demystifying 3D ICs: the pros and cons of going vertical. *Design & Test of Computers, IEEE 22*, 6 (2005), 498–510.

[14] DENG, Y., AND MALY, W. 2.5 D system integration: a design driven system implementation schema. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)* (2004), IEEE, pp. 450–455.

[15] FOGGIA, P., SANSONE, C., AND VENTO, M. A performance comparison of five algorithms for graph isomorphism. In *Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition* (2001), pp. 188–199.

[16] GARY, M., AND JOHNSON, D. Computers and intractability: A guide to the theory of np-completeness, 1979.

[17] HICKS, M., FINNICUM, M., KING, S. T., MARTIN, M. M., AND SMITH, J. M. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *Proceedings of the IEEE Symposium on Security and Privacy* (2010), IEEE, pp. 159–172.

[18] IRVINE, C. E., AND LEVITT, K. Trusted hardware: Can it be trustworthy? In *Proceedings of the 44th Annual Design Automation Conference* (2007), ACM, pp. 1–4.

[19] KING, S., TUCEK, J., COZZIE, A., GRIER, C., JIANG, W., AND ZHOU, Y. Designing and implementing malicious hardware. In *Proceedings of the 1st USENIX Workshop on Large-scale Exploits and Emergent Threats* (2008), USENIX Association, pp. 1–8.

[20] KUON, I., AND ROSE, J. Measuring the gap between fpgas and asics. In *Proceedings of the 2006 ACM/SIGDA 14th international Symposium on Field programmable gate arrays* (2006), ACM, pp. 21–30.

[21] LAU, J. H. TSV interposer: The most cost-effective integrator for 3D IC integration. *Chip Scale Review* (2011), 23–27.

[22] MICCIANCIO, D., AND GOLDWASSER, S. *Complexity of Lattice Problems: a cryptographic perspective*, vol. 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[23] MICROSEMI. Microsemi ProASIC3 FPGA security overview, 2012. Available from `www.microsemi.com/documents/`.

[24] RAJENDRAN, J., PINO, Y., SINANOGLU, O., AND KARRI, R. Security analysis of logic obfuscation. In *Proceedings of the 49th Annual Design Automation Conference* (2012), ACM, pp. 83–89.

[25] RAJENDRAN, J., SINANOGLU, O., AND KARRI, R. Is split manufacturing secure? In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2013), IEEE, pp. 1259–1264.

[26] ROY, J. A., KOUSHANFAR, F., AND MARKOV, I. L. Epic: Ending piracy of integrated circuits. In *Proceedings of the conference on Design, Automation and Test in Europe* (2008), ACM, pp. 1069–1074.

[27] SENTOVICH, E. M., SINGH, K. J., MOON, C., SAVOJ, H., BRAYTON, R. K., AND SANGIOVANNI-VINCENTELLI, A. Sequential circuit design using synthesis and optimization. In *Proceedings of the IEEE International Conference on Computer Design (ICCD)* (1992), IEEE, pp. 328–333.

[28] SKOROBOGATOV, S., AND WOODS, C. Breakthrough silicon scanning discovers backdoor in military chip. *Cryptographic Hardware and Embedded Systems–CHES* (2012), 23–40.

[29] SORENSSON, N., AND EEN, N. Minisat v1. 13-a sat solver with conflict-clause minimization. *SAT 2005* (2005), 53.

[30] SWEENEY, L. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 05 (2002), 557–570.

[31] TEHRANIPOOR, M., AND KOUSHANFAR, F. A survey of hardware trojan taxonomy and detection. *Design & Test of Computers, IEEE 27*, 1 (2010), 10–25.

[32] VALAMEHR, J., TIWARI, M., SHERWOOD, T., KASTNER, R., HUFFMIRE, T., IRVINE, C., AND LEVIN, T. Hardware assistance for trustworthy systems through 3-D integration. In *Proceedings of the 26th Annual Computer Security Applications Conference* (2010), ACM, pp. 199–210.

[33] WAKSMAN, A., AND SETHUMADHAVAN, S. Silencing hardware backdoors. In *Proceedings of the IEEE Symposium on Security and Privacy* (2011), IEEE, pp. 49–63.

[34] ZHOU, B., AND PEI, J. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE)* (2008), IEEE, pp. 506–515.

A k-SECURITY-DEC is NP-hard

In this section, we provide outlines of the proofs that underlie our assertion in Section 3 that k-SECURITY-DEC is **NP**-hard under polynomial-time Turing, or Cook, reductions [5]. Such reductions work the following way. Suppose we want to reduce problem A to B. We show that if we have an oracle for B, then $A \in \mathbf{P}$.

Such reductions are unlikely to be as strong as Karp-reductions [5], that are customarily used to show **NP**-hardness. Indeed, the Karp-reduction is a special case of the Cook-reduction, and some of our reductions below are Karp-reductions. Nevertheless, the existence of a Cook-reduction from a problem that is **NP**-hard is evidence of intractability [22]. In particular, in the above example, if A reduces to B, then if $B \in \mathbf{P}$, then $A \in \mathbf{P}$.

Recall from Section 3 that k-SECURITY-DEC is the following decision problem. Given as input $\langle G, E', k\rangle$ where $E' \subseteq E[G]$, does lifting the edges in E' give us k-security? We show that k-SECURITY-DEC is **NP**-hard in three steps. First, we show that SUB-ISO-SELF (defined below) is **NP**-hard. We then reduce SUB-ISO-SELF to GATE-SUBISO (see Section 3), thereby showing that GATE-SUBISO is **NP**-hard. Finally, we reduce GATE-SUBISO to k-SECURITY-DEC.

All graphs we consider are directed, acyclic (DAGs). Thus, all subisomorphisms we consider are for the special case that the graphs are DAGs. It turns out that the subgraph isomorphism problem is **NP**-hard for even the restricted case, SUB-ISO-9, below.

Definition 6 (SUB-ISO-9)**.** *SUB-ISO-9 is the following special case of the subgraph isomorphism problem. Given as input $\langle G, H\rangle$ where G is a DAG and H is a directed tree, SUB-ISO-9 is the problem of determining whether there exists a subgraph of G that is isomorphic to H.*

SUB-ISO-9 is known to be **NP**-hard [16].

Definition 7 (SUB-ISO-SELF)**.** *Given as input $\langle G, H\rangle$ such that G is a DAG and H is obtained from G by removing the edges in a set $E' \subseteq E[G]$, SUB-ISO-SELF is the problem of determining whether there exists a subgraph isomorphism ϕ from G to H that is not the identity mapping.*

Theorem 2. *SUB-ISO-SELF $\in$ **NP**-hard.*

Note that the above theorem is not qualified that it is under Cook-reductions. This is because we have a Karp-reduction from SUB-ISO-9 ro SUB-ISO-SELF. The reduction proceeds in several steps. First, we show that SUB-ISO-9 restricted to the case that $|V[G]| = |V[H]|$ leaves the problem **NP**-hard. We do this by first observing that for any prospective instance $\langle G, H\rangle$ of SUB-ISO-9, we can assume that $|V[H]| \leq |V[G]|$. We simply add $|V[G]| - |V[H]|$ vertices to H.

Then, we show that if we add the further restriction that G and H are strongly connected (i.e., every vertex reachable from every other vertex), the problem is still **NP**-hard. For this reduction, we first check whether the two graphs are strong connected. If not, we introduce a new vertex of a colour distinct from every vertex in the graphs which has an edge to and from every other vertex.

We then show that SUB-ISO-SELF is **NP**-hard as follows. We introduce into G an exact copy of H that is disjoint from G. We call this new graph G', and the subgraph of G' that is the copy of H, H'. We further restrict H and H' to not have any automorphisms. To achieve this, we introduce $|V[H]|$ vertices each of a distinct colour, associated with each $u \in V[H]$. Call this vertex v_u. We connect u and v_u with an edge. We do the same in H'. We also add a subgraph G'' to H which has $|V[G]|$ vertices and no edges. (This guarantees that the new subgraph is subgraph isomorphic to G.) We call this new graph H''.

We use the same technique as above of adding coloured vertices to ensure that G (within G') and G'' in H'' are not automorphic. Finally, we connect every new vertex added above to the vertices of G, to every original vertex of H', and every new vertex added to H' to every original vertex of G. We do the same in H''. We now are able to show that $\langle G, H\rangle$ is a true instance of SUB-ISO-9 if and only if $\langle G', H''\rangle$ is an instance of SUB-ISO-SELF.

Theorem 3. *GATE-SUBISO $\in$ **NP**-hard under Cook-reductions.*

Recall that GATE-SUBISO comprises those instances $\langle G, E', u, v\rangle$, where, if H is produced from G by removing the edges in E', and u, v are distinct vertices in G (and therefore H), there is a subgraph isomorphism from G to H that maps u to v. In our reduction, we assume that we have an oracle for GATE-SUBISO. We simply invoke it for every pair of vertices $u, v \in G$. If any of them is true, then we know that $\langle G, H\rangle$ is a true instance of SUB-ISO-SELF. Otherwise, it is not.

Theorem 4. *k-SECURITY-DEC $\in$ **NP**-hard under Cook-reductions.*

We Karp-reduce GATE-SUBISO to k-SECURITY-DEC. Let $\langle G, E', k\rangle$ be a prospective instance of k-SECURITY-DEC, and H is produced from G by removing the edges in E'. We first ensure that every vertex other than u is 2-secure. We do this by introduce a new vertex for every vertex other than u that has exactly the same connectivity. Then, in G, we introduce a new vertex of a completely new colour and attach it to u and v. We include the edge between v and this new vertex in E'. Call the G so modified G'', and the new set of edges E''. We can now show that $\langle G'', E'', 2\rangle$ is a true instance of k-SECURITY-DEC if and only if $\langle G, E', u, v\rangle$ is a true instance of GATE-SUBISO.