TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/10 pp62–71
DOI: 10.26599/TST.2019.9010042
Volume 26, Number 1, February 2021

A Multi-Flow Information Flow Tracking Approach for Proving Quantitative Hardware Security Properties

Yu Tai, Wei Hu*, Lu Zhang, Dejun Mu, and Ryan Kastner

Abstract: Information Flow Tracking (IFT) is an established formal method for proving security properties related to confidentiality, integrity, and isolation. It has seen promise in identifying security vulnerabilities resulting from design flaws, timing channels, and hardware Trojans for secure hardware design. However, existing IFT methods tend to take a qualitative approach and only enforce binary security properties, requiring strict non-interference for the properties to hold while real systems usually allow a small amount of information flows to enable desirable interactions. Consequently, existing methods are inadequate for reasoning about quantitative security properties or measuring the security of a design in order to assess the severity of a security vulnerability. In this work, we propose two multi-flow solutions — multiple verifications for replicating existing IFT model and multi-flow IFT method. The proposed multi-flow IFT method provides more insight into simultaneous information flow behaviors and allows for proof of quantitative information flow security properties, such as diffusion, randomization, and boundaries on the amount of simultaneous information flows. Experimental results show that our method can be used to prove a new type of information flow security property with verification performance benefits.

Key words: hardware security; Information Flow Tracking (IFT); multi-flow IFT; security property

1 Introduction

Over the past decade, numerous security attacks have been targeting the underlying hardware of computing systems by exploiting the security vulnerabilities which remain undetected at the design and fabrication stages. Despite the growing security threats, the current hardware design flow does not take into account security properties and hence guaranteeing such properties remains an open problem in most real world scenarios or still heavily relies on code review. The ever increasing complexity of digital hardware design has made manual inspection for security flaws almost infeasible, while the distributed nature of the hardware supply chain has raised the possibility of the existence of malicious vulnerabilities intended by an untrusted party. Although various tools have been developed for functional testing and verification over the past decade, there is a huge demand for alike tools designated to detect and debug security vulnerabilities.

Information Flow Tracking (IFT) techniques have been widely investigated to create new secure hardware design tools for detecting security defects or enforcing security policies, such as non-interference[1]. Such tools mainly consist of two elements: a security lattice for defining the allowed flows among different security classifications, and a model for tracking data with security labels indicated by the lattice. The security lattice can establish policies, such as restricting the flow of sensitive information to publicly accessible locations or the unauthorized modification of trusted memory. The IFT model then defines how data with different

• Yu Tai, Wei Hu, and Dejun Mu are with the School of Cybersecurity, Lu Zhang is with the School of Automation, Northwestern Polytechnical University, Xi'an 710072, China. E-mail: {taiyu, weihu, mudejun, willvsnick}@mail.nwpu.edu.cn.
• Ryan Kastner is with the Department of Computer Science and Engineering, University of California, San Diego, CA 92093, USA. E-mail: kastner@ucsd.edu.
* To whom correspondence should be addressed.
Manuscript received: 2019-05-27; accepted: 2019-08-28

security labels propagate through the circuit and hence can be used to prove isolation or detect harmful flows of information. IFT methods have been deployed at various abstraction levels of digital hardware designs, including the architecture[2], register transfer[3,4], and gate level[5–7]. However, existing IFT methods typically take a qualitative approach and only provide a binary answer (*yes* or *no*) to security. As an example, it may indicate two different Rivest-Shamir-Adleman (RSA) implementations both leak the key through a timing channel, but cannot tell which one leaks more (or less secure).

The lack of effective tool support for hardware security calls for the study of more convoluted security properties, such as quantified metrics and comparative analysis. For example, existing IFT methods can easily detect a flow from the secret key to the ciphertext, while none of them can quantify the number of information flows or indicate whether the flow is severe enough to allow recovering the key. This requires a finer grained measurement and assessment of information flows. Shannon[8] pioneered in the notion of information theory, which provides the mathematical foundation for measuring security. Statistical and information theoretical metrics, such as variance, entropy, and mutual information have been employed for quantifying security, such as attackability, randomness, and information leakage[9–11]. These theories are also employed to analyze side channel measurements, such as timing delay and power consumption to quantify the amount of leakage[12] or to evaluate side-channel leakage[13]. However, these quantitative metrics are usually estimated from discrete data samples collected from functional verification. They cannot be used for formal proof of quantitative security properties, e.g., tight boundaries on the amount of information flows.

In this work, we demonstrate a new hardware IFT technique that can simultaneously track multiple flows of information by defining multi-bit security labels, designing novel label propagation rules, and deriving multi-flow IFT models. We show how this technique can be used to analyze a new type of security property, quantify the flow of information in hardware designs, and search for attack vectors that can cause a security property violation and pinpoint design flaws. Specifically, this paper makes the following contributions.

- Proposing two multi-flow IFT solutions for measuring simultaneous information flows;
- Providing a method for testing and formal verification of quantitative information flow security properties;
- Providing a technique for searching for attack vectors and identifying the source of information flows;
- Presenting experimental results to demonstrate the verification performance benefits of the proposed technique in proving quantitative security properties.

The remainder of the paper is organized as follows. We briefly review related work in Section 2. Section 3 discusses about different types of information flow security properties and motivates for multi-flow IFT methods. In Section 4, we propose two different approaches for multi-flow tracking and a method for verifying multi-flow security properties. Section 5 presents security verification and performance evaluation results. We conclude the paper in Section 6.

2 Related Work

Hardware IFT techniques have been used to enforce information flow security properties related to confidentiality, integrity, and isolation for secure hardware design[3–5].

Gate Level Information Flow Tracking (GLIFT) is a fine grained IFT technique that allows for precise measurement of all logical flows through Boolean gates. GLIFT has been employed to identify and eliminate timing channels in standard bus interfaces[14], prove isolation of IP blocks in SoC systems[15], craft verifiably information flow secure architectures[16], investigate timing-based security violations[7], and detect a certain class of hardware Trojans[17]. Despite of the success of GLIFT, researchers move to a higher level of abstraction to accelerate security verification for better performance, which is a typical tradeoff in hardware design.

Several IFT methods targeting the Register Transfer Level (RTL) have been introduced over the past few years. Caisson[18] is a novel Hardware Description Language (HDL) with a static type system dedicated to security analysis. Using Caisson, the designer is required to define a security label for each variable, which is then verified by the type system. Sapper[19] improves upon Caisson by designing a dynamic type system that allows the designer to either track the flow of information or enforce certain policies. However, Sapper and Caisson are both finite state-based languages which make their adaptation in the general hardware design flow intricate. Furthermore, the label propagation rules employed in these methods are overly conservative

and overestimate the actual flow. SecVerilog[3] allows the designer to describe his/her own label propagation rules for different operations through the definition of dependable and mutable type functions, which can be verified by the type system. Hence, the designer is responsible for labeling data values and defining the tracking rules, an arduous task that preferably would be automated. Another IFT method at the RTL augments additional Coq semantic model for Verilog to track and prove the secrecy of internal sensitive data[20]. A novel code-space randomization scheme has been presented to defense against code reuse attacks that can hijack the control information flow of the victim application[21]. RTLIFT[4] allows for automatically generating the IFT logic using precise flow tracking rules. The RTLIFT logic can be analyzed by the available EDA tools, eliminating the need for designing a new type system. A property specific approach for information flow security was proposed to accelerate security verification and restrict potential security violations to quickly pinpoint hardware security vulnerabilities[22].

In all these previous work, a binary answer (*yes* or *no*) is provided regarding the flow of information between different design elements, which in turn restricts the capabilities of these methods. A binary answer cannot be used for assessing the severity of the information flow, e.g., the amount of leakage or performing comparative security analysis, e.g., if one implementation leaks more information than another.

Quantitative information flow analysis provides a method to ascertain the security of a system in a finer granularity. Gray III[10] formalized quantitatively an applied flow model that relates non-interference to the maximum rate of flow between variables. Clark et al.[9] restricted the leaked information using different information theoretic measures. Heusser and Malacaria[11] introduced quantitative information analysis for C code and show the information leakage vulnerabilities in the Linux Kernel. Mao et al.[12] employed information theoretic measures to quantify the amount of timing information flow that can be exacted from runtime measurements of cryptographic hardware. However, these methods are all based on the estimation of statistical or information theoretic parameters using discrete data samples from functional verification. None of them takes a formal approach that allows for formal proof of security properties.

In this work, we enhance hardware IFT methods to track multiple flows simultaneously by defining multi-bit security labels and constructing multi-flow IFT models, which will allow understanding simultaneous information flow behaviors and analyzing a new type of security properties.

3 Information Flow Security Properties

Information flow security properties can be categorized into qualitative and quantitative.

3.1 Qualitative information flow security properties

Non-interference is a strong qualitative security property for enforcing data confidentiality and integrity. It states that `high` information should never flow to a `low` portion of the design. For example, consider a cryptographic core where the key should never flow to the ciphertext ready signal. We can label the key as `high` and check if the ready signal is always `low`. The following describes such a security property.

---

set key = `high`,
set others = `low`,
assert ready == `low`.

Non-interference is a very strict qualitative security property. Most existing IFT methods are built upon the non-interference model. These methods are effective for enforcing security properties such as there is either *never* or *always* a flow. They will indicate a potential security threat whenever an unintended information flow is detected.

However, practical systems sometimes allow a small amount of information flow to enable interactions between different design components. Qualitative information flow security properties cannot be used to enforce scenarios where information is allowed to flow but only in a secure manner. Consider a password checker that takes a 64-bit phrase and outputs logical `1` when the input is the correct password otherwise logical `0`. A qualitative information flow security property will indicate a security violation due to the flow from the password to the status output. However, it is perfectly fine to allow such a small amount of information flow.

3.2 Quantitative information flow security properties

Diffusion, confusion, and randomization represent three important security properties for assessing the security of information flows.

Diffusion is a quantitative security property that can

be verified using a Satisfiability Modulo Theories (SMT) solver. It requires that if we change a single bit of the message, statistically half of the ciphertext bits should flip. In the following, we mark the w-th bit of the message as `high` and check if at least half of the ciphertext bits will be `high`, where L is the length of the ciphertext in bits and cipher_t is the security label of the ciphertext.

set message[w] = `high`,
set others = `low`,
assert $\sum$ (cipher_t[j] == `high`) $> L/2,\ j \in [0, L-1]$.

Confusion is another important quantitative security property which requires that each ciphertext bit should depend on several parts of the key. Assume that we have a security label $*\_i$ that can track the information flows from each individual key bit, the following example security property requires that at least N key bits should flow to every single bit of the ciphertext. This can be formalized as Eq. (1) and checked using an SMT solver,

$$\exists\, \text{message} \in \{0,1\}^L,\ \forall\, \text{key} \in \{0,1\}^L,$$
$$\sum \text{cipher\_}i\,[j] > N,\ j \in [0, L-1] \qquad (1)$$

Randomization is a quantitative security property similar to confusion, which can be used to evaluate the strength of a masking mechanism. It requires that a critical variable is at least protected by N bits of the random number. Assume that we have a security label $*\_i$ that can track the information flows from every single bit of the random number, the following example security property requires that at least N bits of the random number should flow to each bit of var,

$$\exists\, \text{message}, \in \{0,1\}^L,\ \forall\, \text{rand} \in \{0,1\}^L,$$
$$\sum \text{var\_}i\,[j] > N,\ j \in [0, L-1] \qquad (2)$$

In the following, we will propose two multi-flow IFT approaches for simultaneously tracking the information flows from each individual bit of a variable and enforcing quantitative information flow security properties.

4 Multi-Flow IFT Approaches

4.1 Multi-flow information flow tracking

Existing hardware IFT models only allow a data object to be associated with a single security class at any time. The change in its security classification precisely indicates a flow between the old and updated security classes. However, these models are inadequate in modeling simultaneous information flow scenarios among multiple security classes.

As an example, cryptographic cores usually introduce confusion and diffusion to increase security. This requires that all secret data bits are fully and evenly mixed. It is usually desirable that all bits of the secret key should flow to all bits of the ciphertext. In another case, ideally all bits of the random number should protect (flow to) some critical variable in order to guarantee the strength of a masking mechanism. In such cases, we need a multi-flow IFT model that can track the flow of multiple data bits simultaneously. This model can precisely measure how many bits are affecting (flowing to) a target bit at the same time and allow quantifying the amount of information flow. We present two different solutions for multi-flow IFT in Sections 4.2 and 4.3.

4.2 Label encoding and mapping

To track multiple information flows simultaneously, we need to expand the security label used by existing hardware IFT methods to track the propagation of every single input data bit. This requires linear scale expansion in label width. After the expansion, a simple two-level security lattice `low` $\sqsubset$ `high` will be adequate for security classification. Thus, there still can be significant benefits in scalability as compared to introducing a complex security lattice, which will result in exponential scale increase in complexity of the IFT model.

For a better understanding of the multi-flow security label, consider a four-bit signal A. Table 1 shows the security labels under different IFT models. Under the two-level hardware IFT method, we have the taint labels $A_t[w] \in \{0,1\}$, where $0 \leqslant w \leqslant 3$. Thus, A_t is four-bit wide. The multi-level IFT method allows a finer classification of security labels. Each individual bit in A can take a different label, we have $A_t[w] \in \{00, 01, 10, 11\}$, which can be encoded with two binary bits. Therefore, A_t is eight-bit wide for the multi-level IFT method. By comparison, the multi-flow IFT method uses one-hot encoding in order to track the flow of individual bits. The taint labels $A_i[w] \in \{0000, 0001, 0010, 0100, 1000\}$ (we use a different notation A_i here). Here, 0000 corresponds to the case when $A_t[w] = 0$. The multi-flow model requires 16 bits for security labels.

Now we define a rule for mapping the security labels

Table 1 Encoding methods for security labels under different IFT models.

Model	Signal	Encoding code	Label width
Two-level	$A_t[w]$	0, 1	1 bit
Multi-level	$A_t[w]$	00, 01, 10, 11	2 bits
Multi-flow	$A_i[w]$	0000, 0001, 0010, 0100, 1000	4 bits

to multi-flow. Given an n-bit signal B, we need n-bit security labels in one-hot encoding for multi-flow tacking. Let B_t and B_i be the security labels for existing two-level and multi-flow IFT methods, respectively. Equation (3) shows a possible method to encode the multi-flow security labels, where $<<$ is the left shift operator.

$$B_i[w] = \begin{cases} 0000, & \text{when } B_t[w] = 0; \\ 0001 << w, & \text{when } B_t[w] = 1 \end{cases} \tag{3}$$

4.3 Multi-flow IFT model

Hardware IFT models define the rules for label propagation. These models can precisely measure the existence of a flow by considering the effect of values in label propagation. Take the two-input AND gate (AND-2) as an example. Let A, B, and O be its inputs and output, respectively. Use A_t, B_t, and O_t to denote their security labels under two-level IFT methods. The precise label propagation policy for AND-2 can be formalized as Eq. (4),

$$O_t = A \cdot B_t + B \cdot A_t + A_t \cdot B_t \tag{4}$$

Equation (4) indicates that when A is logical `1` and B is `high`, or B is logical `1` and A is `high`, or both A and B are `high`, the output will be `high`. It models if there is any `high` information flow from either A or B to the output. However, when there is indeed a flow, it does not tell which input pattern caused the output to be `high`. Nor does it reveal how many `high` inputs flow to the output at the same time. Understanding the source of a harmful information flow can be beneficial for pinpointing a security vulnerability. To do this, we need a label propagation policy that tracks the flows of information from multiple variables simultaneously.

Existing work[23] typically tracks the flow of multiple variables using an imprecise model as shown in Fig. 1b. Here, A and B are the original inputs of AND-2; A_i, B_i, and O_i are the multi-flow security labels. This method simply takes the logic OR of all input multi-flow labels as the output label. This doubtlessly will track the flow of all input variables simultaneously. However, it can be overly conservative since it does not take into account the effect of logic values in label propagation. As a simple example, when A is `(low, 0)` and B is `high`, existing hardware IFT model given in Eq. (4) indicates there is no flow of `high` information (i.e., O_t = `low`). Thus, the multi-flow label should be all zeros. However, the multi-flow model shown in Fig. 1b will still indicate a `high` information flow in this case.

Another possible solution for tracking multiple bits is to replicate an existing hardware IFT model (e.g., GLIFT or RTLIFT) and then run multiple verifications for the models replicated as shown in Fig. 1c. However, this will introduce additional verification performance overheads.

To eliminate such false positives and avoid running multiple verifications, we propose a technique to reset the multi-flow label when there is no `high` flow. Specifically, when the precise hardware IFT model (e.g., GLIFT or RTLIFT) indicates there is no `high` flow, we downgrade the multi-flow label to all zeros, i.e., `low`. Figure 1d shows such a technique.

Table 2 shows the number of information flows for outputs of the IWLS *alu2* benchmark measured using the multi-flow tracking models shown in Figs. 1b and 1d,

Table 2 Number of information flow for outputs of the IWLS *alu2* benchmark measured using imprecise and precise multi-flow tracking logic.

Output	Imprecise	Precise	False positive (%)
k	4 194 296	4 053 784	3.47
l	5 242 870	4 792 260	9.40
m	1 048 574	1 048 574	0.00
n	1 048 574	786 430	33.33
o	5 242 870	3 984 738	31.57
p	2 097 148	1 835 004	14.29

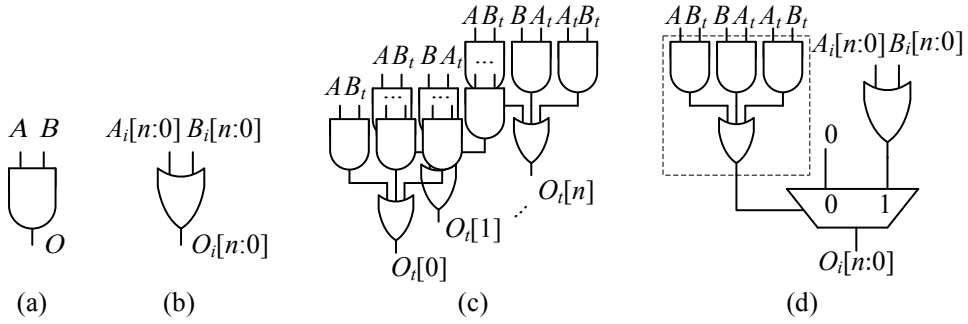


Fig. 1 Multi-flow tracking logic for AND-2. (a) Two-input AND gate. (b) Conservative multi-flow tracking logic for AND-2. (c) Multi-flow tracking logic through IFT model replication. (d) Precise multi-flow tracking logic for AND-2.

respectively. k, l, m, n, o, and p are the outputs of the IWLS *alu2* benchmark. We can see that the percentage of false positives for the imprecise tracking logic shown in Fig. 1b can be over 30% percent for certain outputs. Unlike existing hardware IFT methods, where false positives are safe, false positives in multi-flow tracking may indicate non-existent confusion or diffusion and thus allow an insecure design to be verified as secure.

To derive the multi-flow tracking logic for other Boolean gates, we only need to replace the precise hardware IFT logic for AND-2 in the dotted rectangle in Fig. 1d with the corresponding IFT logic for that gate. The multi-flow tracking logic for an invertor will automatically simplify to a direct connection. This is because an invertor always propagates the security label as it is.

4.4 Multi-flow tracking logic generation

To understand multi-flow behaviors of hardware designs, the first step is to create multi-flow tracking logic for digital circuits. This can be done in a constructive manner similar to technology mapping. We first synthesize the hardware design into design netlist consisted of a small set of logic primitives (e.g., AND, OR, and Invertor). Deriving multi-flow tracking logic for smaller primitives has significantly lower complexity. With the multi-flow tracking logic for the logic primitives, we can then discretely map the primitive gates in the design netlist to its corresponding multi-flow tracking logic. Using this constructive approach, multi-flow tracking logic for large hardware design can be generated in polynomial time. Figure 2 shows such a constructive method.

5 Experimental Results

In this section, we present experimental results. In Section 5.1, we perform simulation analysis to show that multi-flow IFT model reveals the confusion introduced by S-Box. We perform verification analysis to demonstrate how multi-flow IFT models simultaneous information flow behaviors while existing hardware IFT methods cannot in Section 5.2. We perform comparison of security verification performance across different IFT models in Section 5.3.

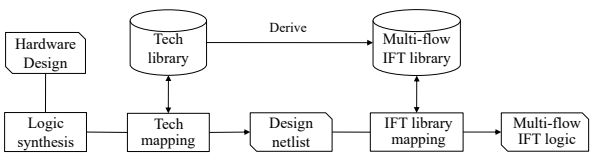


Fig. 2 Constructive method for multi-flow tracking logic generation.

5.1 Simulation analysis

We perform simulation analysis using the S-Box in the Advanced Encryption Standard (AES) cipher. The function under test can be described as $O = \text{sbox}(key \text{ XOR } mes)$, where *key* is a key byte, *mes* is a message byte, and sbox is the substitute bytes operation in AES.

We mark all eight bits of *key* as `high` and encode their multi-flow security labels using the method introduced in Section 4.2. All bits of *mes* are labeled as `low` and thus their multi-flow labels are all zeros. We use linear feedback shift register to generate random vectors as the message and observe the multi-flow labels of the signals in the S-Box model. Figure 3 shows the percentage of signals affected by different number of key bits.

From Fig. 3, the signals affected by no more than two key bits sum up to about 5%. About 8% signals are affected by three key bits. Over 85% signals are affected by at least four key bits, where more than 25% signals are affected by all eight key bits. Our simulation test based upon the multi-flow IFT model reveals the confusion introduced by S-Box, which could not be shown using existing hardware IFT methods.

5.2 Security verification analysis

We perform security verification analysis using an AES core of Differential Power Analysis (DPA) contest from *dpacontest.org*. Figure 4 shows the architecture of the AES design.

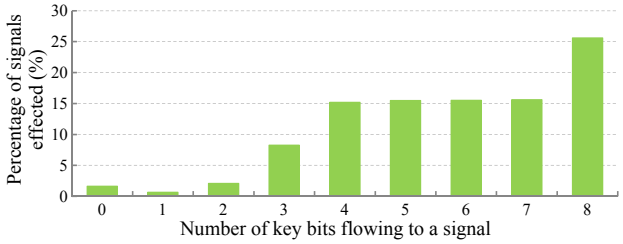


Fig. 3 Percentage of signals affected by different number of key bits from random simulation of AES S-Box using the multi-flow IFT model.

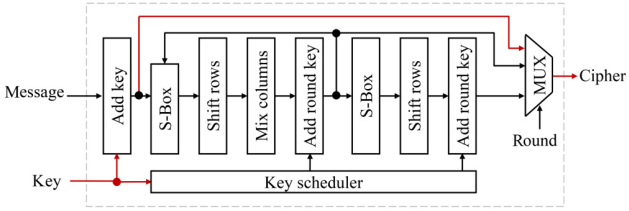


Fig. 4 AES core from DPA contest.

We mark the lowest 16 bits of the key as `high`, i.e., *key_t* = 128'hFFFF. We then use the label encoding and mapping method introduced in Section 4.2 to determine the multi-flow labels for the corresponding key bits. These would be from 16'h0001 to 16'h8000 using our encoding. We perform security verification using both the GLIFT and multi-flow IFT models. We check the following security properties, where cipher_t[w] ($w \in [0, 15]$) is the taint label for the ciphertext under the GLIFT model while cipher_i[w] is the multi-flow label,

assert cipher_t[w] == `high`,
assert $\sum$ cipher_i[w] == 16.

Verification results using GLIFT show that the security property always holds, i.e., the ciphertext is always `high`. However, results indicate that the security property for multi-flow IFT can be violated. After a closer check of the AES design, we find the core contains a security flaw in that it feeds the key *xor* message and intermediate encryption results to the cipher output. These intermediate results leak a significant amount of information about the key. For example, when the message bits are all zeros, the key will be observed at the cipher output directly.

We further perform security verification analysis using a hardware Trojan design that is activated when the plaintext message is 128'h00112233445566778899AABBCCDDEEFF to leak the key. Figure 5 shows the architecture of the Trojan design.

Similarly, We mark the lowest 8 bits of the key as `high`, i.e., *key_t* = 128'hFF and also assign multi-flow labels for the corresponding key bits, which would be from 8'h01 to 8'h80 under our encoding. We perform security verification using both the GLIFT and multi-flow IFT models to check the security properties, where cipher_t[w] ($w \in [0, 7]$) is the taint label for the ciphertext under the GLIFT model while cipher_i[w] is the multi-flow label,

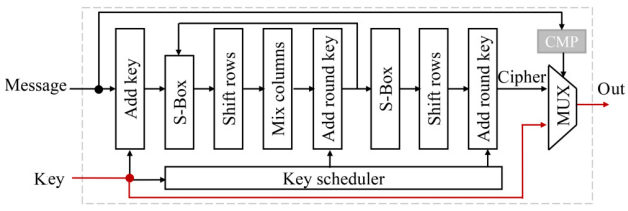


Fig. 5 Hardware Trojan design that is activated when a specific plaintext is observed.

assert cipher_t[w] == `high`,
assert $\sum$ cipher_i[w] == 8.

Verification results using GLIFT show that the security property always holds, i.e., the ciphertext is always `high`. However, results indicate that the security property for multi-flow IFT can be violated when the plaintext message is 128'h00112233445566778899AABBCCDDEEFF, i.e., when the Trojan is triggered.

From the verification results, the multi-flow method captures both the security flaw and Trojan where GLIFT cannot. This is because the multi-flow method can be used to formally verify quantitative security properties while GLIFT only enforces qualitative ones.

5.3 Verification performance analysis

We first compare the verification time of GLIFT and multi-flow IFT models using RSA cores of different key sizes. We use these two IFT models to verify the property that key bits do not flow to the ciphertext ready signal. For the GLIFT model, we mark one key bit as `high` and check if the ready signal will be high when encryption completes. For the multi-flow model, we mark eight key bits as `high` and check if the multi-flow label of ready will be 8'hFF. Figure 6 shows the verification time for different cores.

From Fig. 6, the verification time for both methods will grow with the size of RSA cores. Although the multi-flow method requires more time for running a single security verification, it accounts for eight key bits in a single run. The GLIFT method will need to run multiple similar verifications with different security constraints, which would lead to significantly higher performance overheads. Thus, the multi-flow model may see benefits in performance when verifying quantitative security properties.

We then use several cryptographic functions and *trust-HUB* benchmarks[24] for security verification analysis to

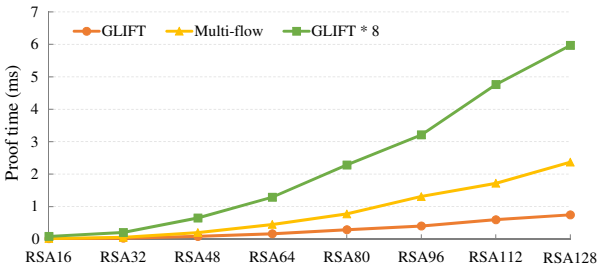


Fig. 6 Verification time for both GLIFT and multi-flow IFT models using RSA cores of different key sizes.

compare their performance and ability to capture harmful information flows. There are some slight differences in the types of security properties that are proved on these two models. Take the *BasicRSA* design for example, we label a signal key bit and would like to check where this key bit will flow to using the GLIFT model; we label eight key bits and track their propagation to each ciphertext bit using the multi-flow model. Table 3 gives the verification reports from Questa Formal.

From Table 3, the multi-flow model tends to take longer verification time. However, it still can be beneficial because this model tracks multiple flows during each proof. GLIFT$\times N$ means it takes N times of GLIFT proofs. Consider the AES-DPA example, it takes the GLIFT model 42 s to prove where a single key bit can flow to. It is important to note that we cannot label multiple key bits for the GLIFT model in this proof. Otherwise, we will not be able to tell which key bit actually propagates to a specific ciphertext bit. To understand the propagation of all the key bits, we need to label a different key bit each time and run multiple verifications under different constraints. For the entire 128-bit key, this will require 128 proofs, which result in a total verification time of $42 \times 128 = 5376$ s. By comparison, the multi-flow method can model simultaneous information flow behaviors. We can monitor the propagation of eight key bits in 67 s. Understanding the propagation of all key bits requires running 16 proofs, which yields to a total proof time of $67 \times 16 = 1072$ s.

It is important to notice that for the presentcipher and RSA-T300 benchmarks, the GLIFT method indicates the key bits always flow to all ciphertext bits while the multi-flow IFT method says no. Here, the multi-flow IFT method more precisely captures the key leakage through intermediate encryption result in present cipher and also the leakage via a hardware Trojan in the RSA-T300 design.

As demonstrated by our tests, the multi-flow IFT method can be more complex than GLIFT. However, it can be used to prove quantified security properties, such as diffusion and confusion. By comparison, the GLIFT and multi-level models can be more efficient in proving qualitative security properties, e.g., no key bit should flow to a given output.

6 Conclusion

In this paper, we propose a multi-flow IFT method for understanding the simultaneous flows of information through hardware designs. The proposed method provides a finer insight into the simultaneous information flow behaviors and allows verifying a wider range of security properties with potential verification performance benefits. The multi-flow method detects both unintentional design flaws and malicious hardware Trojans where qualitative IFT method cannot through formal verification of quantitative security properties.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 61672433) and the Natural Science Foundation of Shaanxi Province (No. 2019JM-244).

References

[1] J. A. Goguen and J. Meseguer, Security policies and security models, in *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1982, pp. 11–20.

[2] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas, Secure program execution via dynamic information flow tracking, in *ACM Sigplan Notices*, vol. 39, no. 11, pp. 85–96, 2004.

Table 3 Formal proof results for *OpenCores* cryptographic functions and *trust-HUB* benchmarks, the ✓symbol means proven while ✗ means counter example found.

Benchmark	Security property	Proof time (s)			Proof result	
		GLIFT	GLIFT$\times N$	Multi-flow	GLIFT$\times N$	Multi-flow
AES-SBox	Confusion property of S-Box	2	16	6	✓	✓
AES-DPA	Key bits flow to all ciphertext bits	42	5376	67	✗	✗
mini-aes	Key bits flow to all ciphertext bits	31	3968	41	✗	✗
BasicRSA	Key bits flow to all ciphertext bits	28	896	74	✓	✓
BasicRSA	Key bits do not flow to ready	25	800	50	✗	✗
present_encryptor	Key bits flow to all ciphertext bits	9	720	6	✗	✗
presentcipher	Key bits flow to all ciphertext bits	14	1120	9	✓	✗
tiny_encryption_algorithm	Key bits do not flow to ready	7	896	8	✓	✓
RSA-T300	Key bits flow to all ciphertext bits	52	1664	53	✓	✗
AES-T100	Key bits do not flow to capacitance	45	5760	70	✗	✗

[3] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, A hardware design language for timing-sensitive information-flow security, in *Proc. of Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, Istanbul, Turkey, 2015, pp. 503–516.

[4] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, Register transfer level information flow tracking for provably secure hardware design, in *Proc. of 2017 IEEE Int. Design, Automation & Test in Europe Conference & Exhibition Conf.*, Lausanne, Switzerland, 2017, pp. 1691–1696.

[5] M. Tiwari, H. M. Wassel, B. Mazloom, S. Mysore, F. T. Chong, and T. Sherwood, Complete information flow tracking from the gates up, *ACM Sigplan Notices*, vol. 44, no. 3, pp. 109–120, 2009.

[6] W. Hu, J. Oberg, A. Irturk, M. Tiwari, T. Sherwood, D. Mu, and R. Kastner, Theoretical fundamentals of gate level information flow tracking, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 8, pp. 1128–1140, 2011.

[7] J. Oberg, S. Meiklejohn, T. Sherwood, and R. Kastner, Leveraging gate-level properties to identify hardware timing channels, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, pp. 1288–1301, 2014.

[8] C. E. Shannon, A mathematical theory of communication, *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[9] D. Clark, S. Hunt, and P. Malacaria, Quantified interference for a while language, *Electronic Notes in Theoretical Computer Science*, vol. 112, pp. 149–166, 2005.

[10] J. W. Gray III, Toward a mathematical foundation for information flow security, *Journal of Computer Security*, vol. 1, nos. 3&4, pp. 255–294, 1992.

[11] J. Heusser and P. Malacaria, Quantifying information leaks in software, in *Proc. of 26th ACM Annual Computer Security Applications Conf.*, Austin, TX, USA, 2010, pp. 261–269.

[12] B. Mao, W. Hu, A. Althoff, J. Matai, Y. Tai, D. Mu, T. Sherwood, and R. Kastner, Quantitative analysis of timing channel security in cryptographic hardware design, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1719–1732, 2018.

[13] L. Zhang, W. Hu, A. Ardeshiricham, Y. Tai, J. Blackstone, D. Mu, and R. Kastner, Examining the consequences of high-level synthesis optimizations on power side-channel, in *Proc. of 2018 IEEE Int. Design, Automation & Test in Europe Conference & Exhibition Conf.*, Dresden, Germany, 2018, pp. 1167–1170.

Yu Tai received the PhD degree from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 2018. He is currently a research assistant with the School of Cybersecurity, Northwestern Polytechnical University, China. His current research interests include hardware security, logic synthesis, and optimization in hardware information flow.

[14] J. Oberg, W. Hu, A. Irturk, M. Tiwari, T. Sherwood, and R. Kastner, Information flow isolation in I2C and USB, in *Proc. of 48th ACM/EDAC/IEEE Design Automation Conference*, New York, NY, USA, 2011, pp. 254–259.

[15] R. Kastner, J. Oberg, W. Hu, and A. Irturk, Enforcing information flow guarantees in reconfigurable systems with mix-trusted IP, in *Proc. of Int. Conf. on Engineering of Reconfigurable Systems and Algorithms*, Las Vegas, NV, USA, 2011, pp. 1–12.

[16] M. Tiwari, J. K. Oberg, X. Li, J. Valamehr, T. Levin, B. Hardekopf, R. Kastner, F. T. Chong, and T. Sherwood, Crafting a usable microkernel, processor, and I/O system with strict and provable information flow security, *ACM SIGARCH Computer Architecture News*, vol. 39, no. 3, pp. 189–200, 2011.

[17] W. Hu, B. Mao, J. Oberg, and R. Kastner, Detecting hardware trojans with gate-level information-flow tracking, *Computer*, vol. 49, no. 8. pp. 44–52, 2016.

[18] X. Li, M. Tiwari, J. K. Oberg, V. Kashyap, F. T. Chong, T. Sherwood, and B. Hardekopf, Caisson: A hardware description language for secure information flow, *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 109–120, 2011.

[19] X. Li, V. Kashyap, J. K. Oberg, M. Tiwari, V. R. Rajarathinam, R. Kastner, T. Sherwood, B. Hardekopf, and F. T. Chong, Sapper: A language for hardware-level security policy enforcement, *ACM SIGPLAN Notices*, vol. 42, no. 4, pp. 97–112, 2014.

[20] Y. Jin and Y. Makris, Proof carrying-based information flow tracking for data secrecy protection and hardware trust, in *Proc. of IEEE 30th VLSI Test Symposium*, Maui, HI, USA, 2012, pp. 252–257.

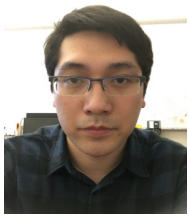
[21] F. Yan and K. Wang, Leakage is prohibited: Memory protection extensions protected address space randomization, *Tsinghua Science and Technology*, vol. 24, no. 5, pp. 546–556, 2019.

[22] W. Hu, A. Ardeshiricham, M. S. Gobulukoglu, X. Wang, and R. Kastner, Property specific information flow analysis for hardware security verification, in *Proc. of IEEE/ACM Int. Conf. on Computer-Aided Design*, San Diego, CA, USA, 2018, pp. 1–8.

[23] B. Mazloom, S. Mysore, M. Tiwari, B. Agrawal, and T. Sherwood, Dataflow tomography: Information flow tracking for understanding and visualizing full systems, *ACM Transactions on Architecture and Code Optimization*, vol. 9, no. 1, pp. 1–26, 2012.

[24] H. Salmani, M. Tehranipoor, and R. Karri, On design vulnerability analysis and trust benchmarks development, in *Proc. of IEEE Int. Conf. on Computer Design*, Asheville, NC, USA, 2013, pp. 471–474.

Wei Hu received the PhD degree from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 2012. He is currently an associate professor with the School of Cybersecurity, Northwestern Polytechnical University, China. His research interests include hardware security, logic synthesis, formal verification, reconfigurable computing, and embedded systems.



Lu Zhang received the BE degree from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 2012, where he is currently pursuing the PhD degree. From January 2016 to January 2018, he was a visiting PhD candidate with the Department of Computer Science and Engineering, University of California, San Diego, CA, USA. His research interests include hardware security, design automation, and embedded systems and optimization.

Dejun Mu received the PhD degree from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 1994. He is currently a professor with the School of Cybersecurity, Northwestern Polytechnical University, China. His current research interests include network information security, application specific chips for information security, and network control systems.

Ryan Kastner received the PhD degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2002. He is currently a professor with the Department of Computer Science and Engineering, University of California at San Diego, San Diego, CA, USA. His current research interests include hardware acceleration, hardware security, and remote sensing. Prof. Kastner is the co-director of the Wireless Embedded Systems Master of Advanced Studies Program. He also co-directs the Engineers for Exploration Program.