MP$\ell\circ$C: Privacy-Preserving IP Verification using Logic Locking and Secure Multiparty Computation

Dimitris Mouris*, Charles Gouert*, and Nektarios Georgios Tsoutsos

{jimouris, cgouert, tsoutsos}@udel.edu
University of Delaware

Abstract. The global supply chain involves multiple independent entities, and potential adversaries can exploit different attack vectors to steal proprietary designs and information. As a result, intellectual property (IP) owners and consumers have reasons to keep their designs private. Without a trusted third party, this mutual mistrust can lead to a deadlock where IP owners are unwilling to disclose their IP core before a financial agreement is reached, while consumers need assurance that the proprietary design will meet their integration needs without compromising the confidentiality of their test vectors. To address this challenge, we introduce an efficient framework called MP$\ell\circ$C that resolves this deadlock by allowing owners and consumers to jointly evaluate the target design with consumer-supplied test vectors while preserving the privacy of both the IP core and the inputs. MP$\ell\circ$C is the first work that combines secure multiparty computation (MPC) and logic-locking techniques to accomplish these goals. Our approach supports both semi-honest and malicious security models to allow users to balance stronger security guarantees with performance. We compare our approach to existing state-of-the-art works that utilize homomorphic encryption across several benchmarks and report runtime improvements of more than two orders of magnitude.

Keywords: Applied Cryptography · Hardware security · Intellectual Property (IP) Verification · IP Piracy · Logic Locking · Secure Multiparty Computation (MPC)

1 Introduction

In the modern integrated circuit (IC) design industry, System on Chip (SoC) solutions have become increasingly popular due to their ability to meet customer design requirements while maintaining a low time-to-market timeline. As a result, the number of IP core vendors and users has increased significantly. To increase profit potential, hardware Intellectual Property (IP) vendors make their IPs reusable by providing design standards and guidelines for use in multiple design layouts to drive sales, leading to a vast array of IP cores being developed across the world.

Collaboration between IP vendors and users is crucial for IP core verification, which is a critical aspect of System on Chip (SoC) design. The IP consumers provide functional requirements to IP vendors, who then design circuits that meet these specifications [1]. The primary objective of IP core verification is to ensure that the IP core operates correctly and satisfies the required specifications. Formal logic verification [2], simulation-based methods [3], and application-specific instruction-set processors [4] are among the most commonly used solutions for IP core verification. While various verification tools such as satisfiability (SAT) solvers [5] are available, simulation methods remain the preferred method.

Unfortunately, trust issues between the parties can arise during the verification process which creates a deadlock. IP vendors are unwilling to disclose their IP until a consumer purchases the netlist as the consumer could manufacture and sell counterfeit devices. Similarly, consumers are hesitant to purchase an IP unless they know it will work for their specific use cases. In some cases, the inputs that consumers want to employ to test the IP may be proprietary and they are unwilling to share this information with IP vendors. For instance, a healthcare provider may want to purchase an IP to be used in hardware for health diagnostics and be sure that the IP will function properly for real patient data, but is unwilling to disclose these test

* The first two authors have equal contributions.

vectors as medical information is highly regulated [6]. Similarly, a company operating a nuclear power plant may want to integrate a third-party IP core into a control system that takes actions based on the uranium enrichment levels. The company will be unwilling to disclose real measurements related to the uranium enrichment process, as this leaks information about the efficiency of the plant. Even if this is not a concern, IP consumers may be wary that the vendor will attempt to forge a correct result with their provided inputs without running the inputs on the actual IP.

The globalized nature of the IP market has also led to an ecosystem consisting of both trusted and untrusted IP owners as well as legitimate and malicious users, making IP theft a significant concern. With vendors prioritizing the functionality of their IPs over security, the problem of IP theft is further exacerbated. As a result, attacks against IPs, such as system-level analysis and reverse engineering, have become very popular. Instead of solely focusing on functionality and runtime performance, it is crucial for the IC supply chain to address security concerns.

Interestingly, recent research has not only focused on traditional security measures but also on using cryptographic techniques like homomorphic encryption (HE) [7, 8] and zero-knowledge proofs (ZKP) [9–11] to enhance the security of transactions in the globalized IC supply chain. For instance, Konstantinou *et al.* [12] and Gouert *et al.* [13] securely outsource the evaluation of public IP netlists to third parties to ensure the confidentiality of the circuit inputs. However, these solutions only protect the input test vectors as homomorphic encryption fails to provide functional privacy; as a result, the actual netlist remains visible to the party that evaluates it. On the other hand, [14–16] preserve the privacy of the netlist using ZKPs and focus both on functional verification and on proving that IPs are free of hardware Trojans. Unfortunately, both HE and ZKP-based solutions are very expensive and impractical for realistic IP cores.

Inspired by recent cryptographic advancements and the efforts for privacy-preserving IP core verification, we propose a practical solution to address the trust issues between IP owners and IP users during the IP core functional verification. Our solution is based on logic locking, a methodology to protect IPs, and secure multiparty computation (MPC), a cryptographic technique that enables multiple parties to jointly compute a function f without revealing their private inputs [17, 18]. In more detail, logic locking protects the circuit by "encrypting" the design using a key, which prevents unauthorized copying or modification of the netlist by adversaries [19]. The encrypted design can only be decrypted using the correct key, which is typically kept secret by the IP owner. Logic locking adds modest overhead in terms of area and latency as new logic must be added to the circuit to enforce correct operation only in the presence of the correct locking key. Our observation is that IP users and IP owners can collaborate and evaluate an IP core while keeping private both the IP (owned by the IP vendor) and the proprietary test vectors (owned by the IP user). Additionally, MPC encompasses a wide range of techniques to ensure that a computation is performed correctly and securely. This technology can be applied to solve a diverse range of problems, from secure data analytics to voting and privacy-preserving machine learning.

1.1 Our Contributions

In this paper, we adopt MPC and logic-locking techniques to create MP$\ell\circ$C: a framework that enables collaboration between IP vendors and users for private evaluation of proprietary test vectors on IPs. Our core idea is to have the IP vendor lock the netlist of the IP and then jointly evaluate the locked IP with the IP user, while keeping both the key and the test vectors private. More specifically, the function f that the two parties evaluate comprises the gates of the locked IP, while the parties' secret inputs are the logic locking key in the case of the IP vendor and the test vectors in the case of the IP consumer. After evaluation, both parties compare the encrypted outputs with the expected outputs and reveal the final pass/fail result.

In this work, we introduce a bespoke compiler from Verilog netlists to "Bristol fashion" MPC circuits [20], which is a serialized format that consists of one gate in each line and guarantees that there are no inter-dependencies between the gate inputs and the outputs from previous gates. Specifically, our compiler ensures that the transformed netlist can be evaluated sequentially. Based on this transformation, our proposed framework wields state-of-the-art semi-honest and maliciously secure MPC protocols to evaluate the logic-locked circuit with private inputs *without the use of a trusted third party (TTP).* For our analysis, we conduct

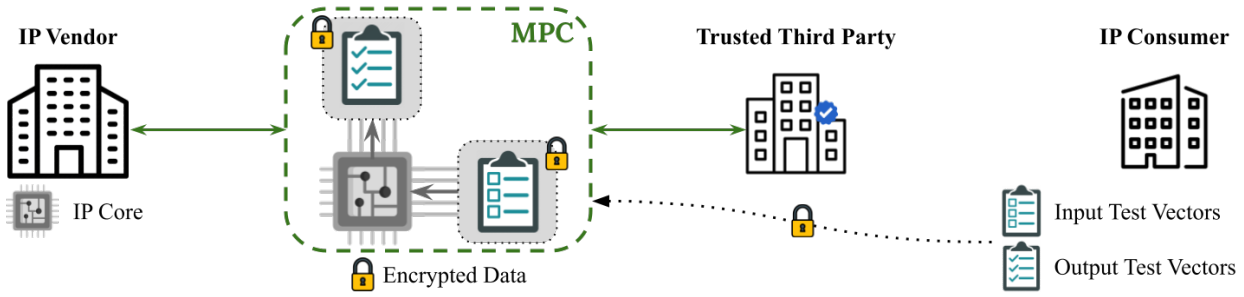

Fig. 1. Strawman Approach. The IP vendor engages in secure computation with a trusted third party (TTP) and they jointly evaluate the IP on the proprietary test vectors of the IP consumer. The test vectors remain private throughout the computation but the IP core is leaked to the TTP.

experiments with ISCAS'85 benchmarks [21] and compare MP$\ell\circ$C against existing solutions that use fully homomorphic encryption as well as a strawman technique that involves integrating a TTP.

1.2 Related Works

Before delving into our framework, we first discuss several recent works for protecting and verifying IP cores.

A recent research direction has focused on outsourcing and evaluating IP designs with encrypted input vectors using homomorphic encryption (HE), which allows performing operations on encrypted data [12]. The authors discuss both a two-party and a three-party case: In the former, the IP user encrypts their vectors and sends them to the IP vendor for homomorphic evaluation. In the latter, the verification is outsourced to a trusted third party that acquires knowledge of the IP. Similarly, [13] converts Verilog netlists into homomorphic circuits and uses encrypted inputs to evaluate them. However, these techniques only protect the privacy of the input test vectors, not the IP itself. Additionally, due to the inherent computational overheads of the underlying cryptographic technique (i.e., HE) these methodologies still remain limited for real-world IP cores. Contrary, as we show in our evaluations, MP$\ell\circ$C protects both the IP and the proprietary test vectors, while achieving scalable evaluation of large IP cores.

A related line of work is based on zero-knowledge proofs (ZKP) and allows IP vendors to prove to IP users various functional and security properties about their IP without revealing any details about it [15,16]. Specifically, these methods can verify that the IP satisfies particular functional properties, meets certain constraints related to the area, performance, and power consumption, as well as security properties (e.g., the IP is free of rarely-activated hardware Trojans [22]). These results are achieved by creating simulators that parse the circuit and examine its behavior based on multiple public input-output test vectors, which are supplied by the IP consumers. However, in this case, the IP consumer has to disclose their test vectors, which is against the objectives of this work. Lastly, similarly to HE-based solutions, ZKP methodologies suffer from high computational overheads.

Garbled circuits [17] enable secure computation between two parties, so that each party does not reveal their inputs to the other party; this is possible by allowing one party to encrypt the circuit and the other party to securely evaluate it. The original garbled circuit scheme can hide the gate types and circuit topology and could be used in our scenario; however, this approach would not be scalable due to high communication overheads [6]. State-of-the-art garbled circuit-based protocols [23] treat different gate types differently, resulting in leakage of the circuit topology. Likewise, existing solutions for private function evaluation can hide the gates of the circuits [6, 24, 25], these works mostly focus on theoretical advancements without concrete implementations.

2 Preliminaries

2.1 Secure Multiparty Computation (MPC)

MPC enables multiple entities to jointly perform a computation without disclosing any individual's private inputs [17, 18, 26–28]. There are many MPC protocols that consider various threat assumptions about how the participants are behaving, resulting in security/performance trade-offs. Most practical protocols assume semi-honest adversaries, meaning that they will follow the protocol specification, as opposed to malicious adversaries. Another consideration is the number of honest participants and how privacy and correctness can be maintained when parties collude. MPC protocols that assume the majority of the parties behave honestly (i.e., honest-majority) typically utilize *secret sharing* as their basic primitive [29, 30]. A (t, n)-secret sharing scheme allows splitting a secret amongst n parties, so that t or more parties can reconstruct the secret while having less than t shares does not reveal anything about the secret. In this case, the parties first represent a target function f as a Boolean or arithmetic circuit and then each party shares its input with the other parties using secret sharing. The parties can now jointly evaluate the circuit gate by gate and finally reconstruct the shares on the output wires, which represent the output of the function f that was encoded as the circuit. On the other hand, assuming a dishonest majority MPC is significantly more challenging and requires specialized solutions such as garbled circuits [17, 31], GMW oblivious transfer [27], cut-and-choose [32], SPDZ [33], MPC in the head [34], and others [35].

2.2 Logic Locking

This method is one of the most popular ways to protect netlists, as it ensures that the intended functionality of the original circuit can only be derived with knowledge of a secret logic locking key [19]. For example, this can be achieved by introducing special *key gates* in the existing circuit that take as input an internal wire and one bit of the logic locking key, as shown in Figure 2 (`XOR` gates highlighted in green). Typically these gates take the form of `XOR`/`XNOR` gates [36], multiplexers [37], or lookup tables [38]. For every wrong bit of the logic locking key, the value of an intermediate wire in the circuit will be flipped, which can result in erroneous outputs. When the correct logic locking key is loaded on the fabricated chip, it is typically stored in tamper-evident memory to protect against malicious attacks [39].

Unfortunately, many earlier logic locking techniques can be broken using oracle-less machine learning attacks [40]. Conversely, provably secure logic locking techniques provide guaranteed security against an attacker with only black-box access [41]. The state-of-the-art ALMOST framework uses security-aware synthesis and ensures that locked designs remain resilient post-synthesis [42]. Although logic-locking techniques protect against overproduction, they typically solve a different problem than the one in this work, as they do not allow vendors to evaluate their IPs on proprietary test vectors (e.g., inputs provided by users). Our goal is to protect both the IP and the test vectors while enabling verification of the IP on the proprietary input-output pairs.

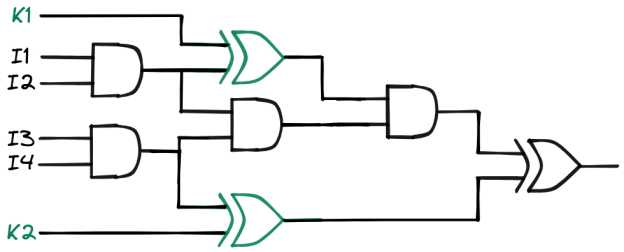

Fig. 2. Logic Locking Example. This circuit illustrates a toy example of a locked circuit with a 2-bit locking key using `XOR` as the key gate type.

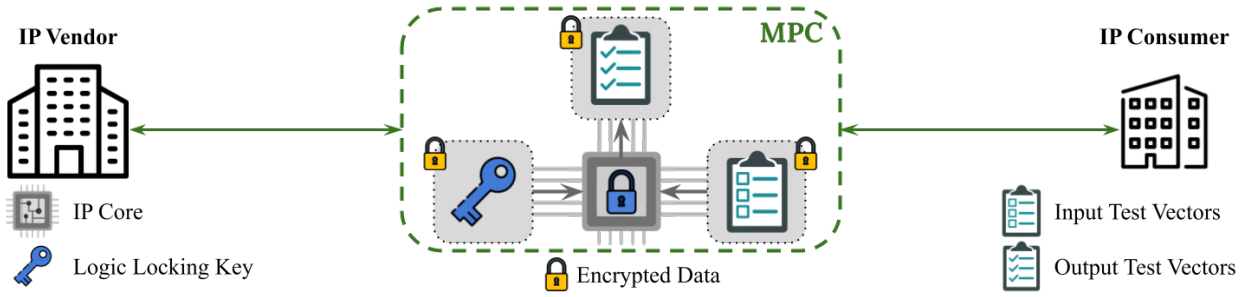

Fig. 3. MP$\ell\circ$C Overview. The IP vendor first locks their IP and then engages in secure computation with the IP consumer. They jointly evaluate the locked IP on their private data; the IP vendor inputs the logic locking key and the IP consumer inputs their proprietary test vectors. Both the logic locking key and the test vectors remain secret throughout the whole computation.

2.3 Threat Model

To formalize our methodology, we now discuss the powers of potential adversaries. MP$\ell\circ$C operates in the two-party computation setting with a *dishonest majority*, meaning that an adversary can corrupt either the IP vendor or the IP consumer, but not both. We assume two distinct threat models that introduce a trade-off between security and performance.

Semi-honest Security Model In this case, we assume that the adversary passively corrupts one of the parties, meaning the corrupt party will faithfully follow the protocol specification but has incentives to eavesdrop the data.

Malicious Security Model In the malicious security model, we assume that the adversary can corrupt one of the parties actively, i.e., the corrupt party may arbitrarily deviate from the protocol specification.

3 Technical Overview

In this work, we combine applied cryptography with hardware security techniques to enable IP vendors and consumers to jointly evaluate a netlist owned by the IP vendor with proprietary test vectors owned by the consumer. More specifically, we adopt logic locking to protect the netlist and secure multiparty computation to jointly evaluate the IP between the consumer and owner, as shown in Figure 3. Next, we offer a technical overview of our MP$\ell\circ$C methodology.

3.1 A Strawman Approach

One way to offer assurance to a consumer that an IP will work for their use case is for the IP owner to share the netlist with the consumer. However, this is only feasible for open-hardware projects as the consumer can plausibly integrate the IP without financial obligations. Alternatively, the consumer can provide the owner with custom test vectors, but these constitute IP on their own as they are often costly to develop or cannot be shared due to the presence of sensitive data.

Both approaches can be strengthened with the use of a third party that is trusted by both the consumer and the owner. In this case, the IP owner will send the netlist to the trusted third-party (TTP) and the consumer will also upload their test vectors. Then, the TTP can directly evaluate the netlist on the provided inputs and share the outputs with the consumer for verification. However, if the third party is not trustworthy, a key concern with this strawman approach involves potential collusion; the third party can easily leak the test vectors to the IP owner or the IP itself to the consumer.

With state-of-the-art cryptographic techniques such as secure MPC, the previously outlined scenario can be further strengthened. Now, the IP owner can share the netlist with the TTP and the consumer generates secret shares of the test vectors to distribute to both the TTP and the owner. The owner and TTP jointly evaluate the IP and send the outputs back to the owner for verification. In this setting, the TTP does not learn the test vectors (unless they actively colludes with the IP owner). This approach is depicted in Figure 1.

3.2 MP$\ell\circ$C: Combining Logic Locking with MPC

MP$\ell\circ$C eliminates the need for a TTP entirely while preserving the confidentiality of both the owner's IP core and the consumer's test vectors. In our proposed methodology, the IP owner relies on provably-secure logic locking techniques to secure their IP and then jointly evaluates the IP with the client using zn MPC scheme. Both the logic locking key, which is owned by the IP owner, as well as the consumer's test vectors are secret-shared with the other party.

In our concrete implementation, we employ the state-of-the-art ALMOST logic locking framework [42], which outputs synthesized Verilog using a 45-nanometer tech library.[1] We emphasize that MP$\ell\circ$C can readily support any other logic locking based methodology that uses standard logic gates as the key gate mechanism. However, it can also extend to lookup table-based methods [38] by exploiting the fact that all lookup tables can be replaced with a subcircuit consisting solely of standard logic cells. In this case, the logic synthesis tool can perform the necessary conversions optimally with a synthesis script. Our MP$\ell\circ$C instantiation employs the Yosys logic synthesis library, so this conversion can be done principally with the `abc -g AND, XOR` command, which instructs the underlying ABC toolchain to map all cells to standard 2-input logic gates and inverters where possible.

Modern MPC implementations that support binary encodings, such as MP-SPDZ [43], typically expect netlists to be defined using Bristol-style circuits. This specification is well-suited for privacy-preserving technologies such as MPC and HE because all logic gates appear in an order that prevents any unresolved dependencies. The preamble of a Bristol-style circuit outlines the number of gates and wires in the circuit, as well as the number of inputs and outputs along with the sizes of each. Each proceeding line describes a single logic gate, specifying the gate type (constrained to `AND`, `XOR`, and `INV`), and wire IDs corresponding to the inputs and outputs.

To convert the synthesized Verilog format to Bristol-style, we first use the Yosys synthesis tool [44] to perform a few necessary transformations. Specifically, MP$\ell\circ$C instructs Yosys to map all cells to the 2-input `XOR` and `AND` gates and the single input `INV` gate in accordance with the supported Bristol gate types. Then, Yosys writes the transformed circuit as an EDIF netlist.

Next, we employ our parser to read the EDIF file and perform a topological sort on the circuit (which is represented as a directed acyclic graph). This is a necessary step as a logic gate can not be evaluated until all input wires have been resolved. Then, MP$\ell\circ$C's compiler traverses the sorted graph and writes each node in the circuit to the output netlist. At last, all net identifiers are converted to be compliant with the Bristol-fashion specification, where the smallest IDs represent input values, the largest IDs represent output wires, and the remaining IDs represent intermediate wires in the circuit.

4 Experimental Results

We report a series of experiments for different logic-locked circuits from the ISCAS'85 benchmark suite [21]. During synthesis, we used the `proc`, `flatten`, `synth`, and `abc -g AND, XOR` flags in Yosys [44] to generate circuits with standard 2-input `AND` and `XOR` logic gates. For logic locking, we utilized the provably secure logic locking ALMOST framework [42] and acquire *ALMOST-protected* netlists. Table 1 depicts the overhead in terms of the number of gates for all ISCAS'85 circuits. For small circuits, we observe that ALMOST results in a high overhead, adding over 4$\times$ more logic gates to the circuit compared to the unlocked baseline. However, for the larger ISCAS circuits, we observe comparatively lower overheads. Specifically, the locked `c5315` circuit

[1] Our framework is universal and can work with any logic-locking technique.

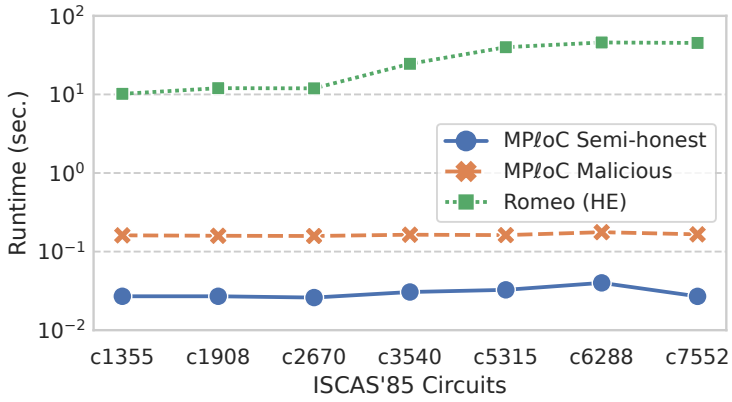

Fig. 4. Total server runtime for selected benchmarks from the ISCAS-85 suite. For the experiments of our framework, we locked the circuits with provably secure logic locking, while for the homomorphic encryption based solution (i.e., Romeo) we used unlocked circuits (since the sole computing party is the IP owner).

is 19% larger than the unlocked version while the locked `c6288` and `c7552` variants have approximately 40% more logic gates. We note that the majority of the added logic consists of `XOR` and `NOT` gates, which run very efficiently with MPC due to optimizations such as free-XOR [45].

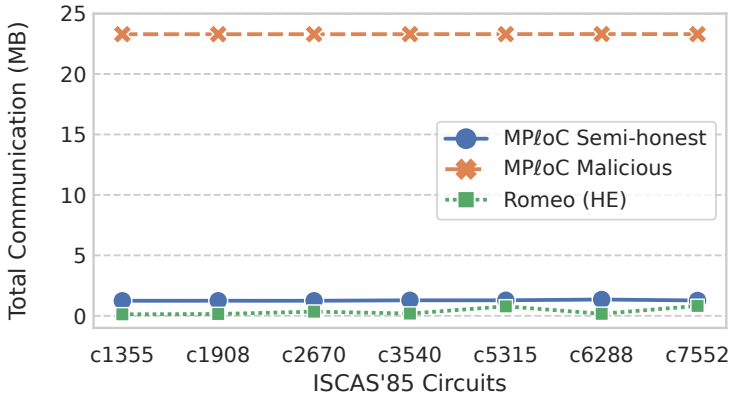

Fig. 5. Total server communication in MB for selected benchmarks from the ISCAS-85 suite. As mentioned, for our experiments, we locked the circuits with provably secure logic locking, while for the homomorphic encryption based solution (i.e., Romeo) we used unlocked circuits (since the sole computing party is the IP owner).

MP$\ell\circ$C relies on the widely used MP-SPDZ [43] framework that implements multiple MPC protocols. We used the default security parameters of MP-SPDZ, namely, a computational security parameter of 128 bits and a statistical security parameter of 40 bits. MP$\ell\circ$C assumes the two-party dishonest majority model, and we experiment with both semi-honest and malicious adversaries. For malicious security, we use the implementation of the MASCOT protocol [46] of MP-SPDZ that allows for secure computation of general arithmetic circuits. For semi-honest security, MP-SPDZ strips out all steps required for malicious security from the MASCOT protocol, resulting in a more efficient protocol. In our analysis, we report both the runtime and the communication overheads for these different security models. Finally, we compare with closely related work in Romeo [13]), which uses homomorphic encryption to protect the test vectors and enable evaluation by the IP owner (or a trusted cloud server). As with our framework, Romeo uses 128 bits of security and also utilizes unlocked circuits as the IP consumer does not need to take part in the circuit evaluation (and therefore does not need the IP to be shared before purchase).

Experimental Setup. We evaluate our framework on two AWS EC2 machines (c5.9xlarge) in the same region, each with 36 vCPUs at 3.60 GHz, while for Romeo, we used one of these instances. Our compiler from EDIF to Bristol-style format [20] is implemented in Python 3.

Table 1. Circuit sizes of unlocked versus ALMOST-protected netlists using a 128-bit locking key.

Circuit	**c1355**	**c1908**	**c2670**	**c3540**	**c5315**	**c6288**	**c7552**
Unlocked	214	293	864	1386	2083	2423	1852
Locked	935	1003	1020	1770	2472	3390	2602

4.1 Runtime Experiments

First, we focus on the runtime of MP$\ell\circ$C and compare it with Romeo in Figure 4 for the ISCAS'85 circuits. MP$\ell\circ$C runs in less than 0.2 seconds for both semi-honest and malicious security for all circuits, achieving almost a constant runtime. We observe that MP$\ell\circ$C in the semi-honest threat model is 4× faster than the malicious case, running all circuits in less than 0.05 seconds. Contrary, Romeo is two orders of magnitude slower and scales linearly with the number of gates of each circuit. For reference, it takes Romeo 10.1 and 45.2 seconds to evaluate `c1355` and `c7552`, respectively. Finally, as reported in [13], an AES circuit is evaluated in approximately 13.5 minutes using Romeo. Nevertheless, our experimental server Romeo requires 20.7 minutes for the AES circuit, which is attributed to a slower processor and adopting larger parameters for 128 bits of security (as opposed to 80 bits in [13]). Conversely, MP$\ell\circ$C evaluates AES with semi-honest security in 0.67 seconds and with malicious security in 53.8 seconds, for a circuit comprising approximately 250 thousand gates. This yields a 1849× improvement for the semi-honest setting and a 23× for malicious.

As a baseline, we also evaluated the unlocked variants of our ISCAS'85 circuits and observe that MP$\ell\circ$C can evaluate them in nearly identical runtime compared to their locked equivalents. This can be mostly attributed to two factors: first, the ALMOST framework does not cause a substantial increase in the circuit size (especially for larger circuits). Second, the underlying MPC library in MP$\ell\circ$C is very efficient and a small increase in the number of gates has a negligible impact on runtime. Even in cases where the circuit becomes considerably larger, the runtime impact remains minimal. For example, the locked `c1908` circuit requires an additional 2 milliseconds while being nearly 4× larger.

4.2 Communication Experiments

Next, we evaluate MP$\ell\circ$C and Romeo in terms of communication overhead and show empirical results in Figure 5. Our framework requires communication between the two parties for evaluation, but for semi-honest security, this is minimal. One of the advantages of using an FHE-based solution like Romeo is that the only communication between the IP consumer and the IP owner is transmitting the encrypted test vectors and getting back the encrypted results. However, we note that FHE solutions also incur additional communication overheads because public key material must be transmitted to the IP owner to facilitate important HE operations. For example, the key-switching operation used during most FHE logic gate evaluations requires a large matrix (referred to as the key-switching key); likewise, FHE noise reduction operations require a bootstrapping key, which is composed of a bit-by-bit encryption of the consumer's secret key.

In total, such FHE public key material consists of approximately 100 MB of data for Romeo. We omit this from Figure 5 as it is a one-time cost, albeit a substantial one. Overall, we observe that both Romeo and the semi-honest MP$\ell\circ$C incur less than 1.2 MBs of communication, while the malicious MP$\ell\circ$C incurs close to 23 MBs.

5 Discussion

MP$\ell\circ$C addresses important considerations related to trust in the global IP supply chain, but the same methodology can be generalized to other fields and address similar privacy issues. Cloud computing is widely used in many industries as it allows organizations to avoid maintaining costly computing infrastructure and instead outsource data to third-party servers that can perform proprietary computations over the inputs. For instance, a cloud service may offer image classification using a proprietary machine learning model and allow users to have their own images classified for a fee [6]. Here, a similar deadlock occurs that mimics those between an IP owner and consumer: an end-user seeks assurances that a cloud service will process their data, yet the cloud service provider is unwilling to share their proprietary algorithms. At the same time, the customer is unwilling to share their sensitive data with the cloud. In this scenario, MP$\ell\circ$C can be used to jointly compute the cloud's proprietary algorithm between the cloud and a user, while ensuring the confidentiality of both the algorithm itself and the user's inputs. If the algorithm is not expressed as a Boolean circuit, high-level synthesis can be used to generate the Verilog input that MP$\ell\circ$C expects.

While our evaluations employ strictly combinational circuits, we note that MP$\ell\circ$C can readily extend to arbitrary sequential circuit evaluations as well. While the Bristol specification does not inherently support sequential circuit elements (such as flip-flops and latches) and has no notion of a clock, sequential behavior can still be emulated with this format. Specifically, unrolling techniques (such as those employed in ROMEO [13]) can be adopted to evaluate the combinational parts of the circuit for each clock cycle. For ROMEO, a key limitation is that neither computing party can check termination conditions (e.g., the status of a ready signal since it is encrypted), so the circuit must be simulated cycle by cycle for the maximum number of iterations required to successfully generate the outputs. Conversely, our MPC techniques can easily allow for intermediate results to be revealed to both parties (such as a termination condition or a ready signal). This would allow for greater efficiency relative to the FHE approach as the sequential evaluation can terminate early if a ready signal is asserted.

6 Concluding Remarks

In this paper, we present MP$\ell\circ$C, the first work that combines secure MPC and logic-locking to allow IP vendors and consumers to jointly evaluate an IP core with proprietary test vectors without revealing either the IP or the test vectors to the other party. Our unique observation is that we can use logic locking to protect a target circuit that is evaluated by both parties and treat both the logic locking key and the test vectors as private inputs for MPC computation. This way, both the IP and the test vectors are protected, enabling privacy-preserving evaluation on proprietary inputs. Notably, MP$\ell\circ$C is flexible and agnostic to the underlying logic locking technique or MPC library. For our evaluations, we employed state-of-the-art logic locking and secure multiparty computation methods to demonstrate the benefits of MP$\ell\circ$C over several combinational benchmarks from the ISCAS'85 suite. Overall, MP$\ell\circ$C offers high scalability and outperforms a state-of-the-art solution based on homomorphic encryption by more than two orders of magnitude.

Acknowledgment

We would like to thank the authors of [42], and in particular Animesh Basak Chowdhury, for providing logic-locked netlists and the valuable feedback about ALMOST.

References

1. A. Deshpande, "Verification of IP-Core based SoC's," in *ISQED*. IEEE, 2008, pp. 433–436.
2. Y. Jin and Y. Makris, "Proof carrying-based information flow tracking for data secrecy protection and hardware trust," in *VTS*. IEEE, 2012, pp. 252–257.
3. G. Moretti *et al.*, "Your Core – My Problem? Integration and Verification of IP," in *DAC*. IEEE/ACM, 2001, pp. 170–171.
4. M. Stadler *et al.*, "Functional verification of intellectual properties (IP): a simulation-based solution for an application-specific instruction-set processor," in *IEEE ITC*, 1999, pp. 414–420.
5. B. Keng and A. Veneris, "Path-Directed Abstraction and Refinement for SAT-Based Design Debugging," *IEEE TCAD*, vol. 32, no. 10, pp. 1609–1622, 2013.
6. A.-R. Sadeghi and T. Schneider, "Generalized universal circuits for secure evaluation of private functions with application to data classification," in *ICISC 08*, ser. LNCS, P. J. Lee and J. H. Cheon, Eds., vol. 5461. Springer, Heidelberg, Dec. 2009, pp. 336–353.
7. C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford university Stanford, 2009.
8. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, Jan. 2020.
9. R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang, "Ligero++: A new optimized sublinear IOP," in *ACM CCS 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM Press, Nov. 2020, pp. 2025–2038.
10. D. Mouris and N. G. Tsoutsos, "Zilch: A Framework for Deploying Transparent Zero-Knowledge Proofs," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3269–3284, 2021.
11. K. Yang, P. Sarkar, C. Weng, and X. Wang, "QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field," in *ACM CCS 2021*, G. Vigna and E. Shi, Eds. ACM Press, Nov. 2021, pp. 2986–3001.
12. C. Konstantinou, A. Keliris, and M. Maniatakos, "Privacy-preserving functional IP verification utilizing fully homomorphic encryption," in *DATE*. EDAA, 2015, pp. 333–338.
13. C. Gouert and N. G. Tsoutsos, "ROMEO: Conversion and Evaluation of HDL Designs in the Encrypted Domain," in *57th ACM/IEEE Design Automation Conference, DAC 2020, July 20-24, 2020*. San Francisco, CA, USA: IEEE, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/DAC18072.2020.9218579
14. D. Mouris and N. G. Tsoutsos, "Pythia: Intellectual Property Verification in Zero-Knowledge," in *57th ACM/IEEE Design Automation Conference, DAC 2020, July 20-24, 2020*. San Francisco, CA, USA: IEEE, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/DAC18072.2020.9218639
15. D. Mouris, C. Gouert, and N. G. Tsoutsos, "Privacy-Preserving IP Verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 7, pp. 2010–2023, 2022.
16. ——, "zk-Sherlock: Exposing Hardware Trojans in Zero-Knowledge," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2022, pp. 170–175.
17. A. C.-C. Yao, "Protocols for secure computations (extended abstract)," in *23rd FOCS*. IEEE Computer Society Press, Nov. 1982, pp. 160–164.
18. Y. Lindell, "Secure Multiparty Computation (MPC)," *Communications of the Association for Computing Machinery*, vol. 64, no. 1, p. 86–96, dec 2020.
19. M. Yasin, J. J. Rajendran, and O. Sinanoglu, *A Brief History of Logic Locking*. Cham: Springer International Publishing, 2020, pp. 17–31.
20. KU Leuven COSIC, "SCALE-MAMBA," https://github.com/KULeuven-COSIC/SCALE-MAMBA, 2019.
21. F. Brglez, "A neural netlist of 10 combinational benchmark circuits," *IEEE ISCAS: Special Session on ATPG and Fault Simulation*, pp. 151–158, 1985.
22. M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.
23. M. Rosulek and L. Roy, "Three halves make a whole? Beating the half-gates lower bound for garbled circuits," in *CRYPTO 2021, Part I*, ser. LNCS, T. Malkin and C. Peikert, Eds., vol. 12825. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 94–124.
24. P. Mohassel and S. S. Sadeghian, "How to hide circuits in MPC an efficient framework for private function evaluation," in *EUROCRYPT 2013*, ser. LNCS, T. Johansson and P. Q. Nguyen, Eds., vol. 7881. Springer, Heidelberg, May 2013, pp. 557–574.
25. M. Rosulek, "Improvements for gate-hiding garbled circuits," in *INDOCRYPT 2017*, ser. LNCS, A. Patra and N. P. Smart, Eds., vol. 10698. Springer, Heidelberg, Dec. 2017, pp. 325–345.

26. D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay - secure two-party computation system," in *USENIX Security 2004*, M. Blaze, Ed. USENIX Association, Aug. 2004, pp. 287–302.
27. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *19th ACM STOC*, A. Aho, Ed. ACM Press, May 1987, pp. 218–229.
28. D. Beaver, "Efficient multiparty protocols using circuit randomization," in *CRYPTO'91*, ser. LNCS, J. Feigenbaum, Ed., vol. 576. Springer, Heidelberg, Aug. 1992, pp. 420–432.
29. A. Shamir, "How to share a secret," *Communications of the Association for Computing Machinery*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
30. A. Beimel and B. Chor, "Universally ideal secret sharing schemes (preliminary version)," in *CRYPTO'92*, ser. LNCS, E. F. Brickell, Ed., vol. 740. Springer, Heidelberg, Aug. 1993, pp. 183–195.
31. A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167.
32. Y. Lindell and B. Pinkas, "An efficient protocol for secure two-party computation in the presence of malicious adversaries," in *EUROCRYPT 2007*, ser. LNCS, M. Naor, Ed., vol. 4515. Springer, Heidelberg, May 2007, pp. 52–78.
33. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *CRYPTO 2012*, ser. LNCS, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, Heidelberg, Aug. 2012, pp. 643–662.
34. Y. Ishai, M. Prabhakaran, and A. Sahai, "Founding cryptography on oblivious transfer - efficiently," in *CRYPTO 2008*, ser. LNCS, D. Wagner, Ed., vol. 5157. Springer, Heidelberg, Aug. 2008, pp. 572–591.
35. D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Priv. Secur.*, vol. 2, no. 2–3, p. 70–246, dec 2018.
36. J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *DATE*. ACM, 2008, pp. 1069–1074.
37. S. M. Plaza and I. L. Markov, "Solving the third-shift problem in ic piracy with test-aware logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.
38. A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE design & Test of computers*, vol. 27, no. 1, pp. 66–75, 2010.
39. M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2015.
40. P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2018, pp. 56–61.
41. S. Patnaik, N. Limaye, and O. Sinanoglu, "Hide and Seek: Seeking the (Un)-Hidden Key in Provably-Secure Logic Locking Techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3290–3305, 2022.
42. A. B. Chowdhury, L. Alrahis, L. Collini, J. Knechtel, R. Karri, S. Garg, O. Sinanoglu, and B. Tan, "ALMOST: Adversarial Learning to Mitigate Oracle-less ML Attacks via Synthesis Tuning," in *57th ACM/IEEE Design Automation Conference, DAC 2023, July 9-13, 2023*. San Francisco, CA, USA: IEEE, 2023, pp. 1–6.
43. M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *ACM CCS 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM Press, Nov. 2020, pp. 1575–1590.
44. C. Wolf, J. Glaser, and J. Kepler, "Yosys - A Free Verilog Synthesis Suite," in *Austrian Workshop on Microelectronics (Austrochip)*, 2013.
45. V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *ICALP 2008, Part II*, ser. LNCS, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds., vol. 5126. Springer, Heidelberg, Jul. 2008, pp. 486–498.
46. M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," in *ACM CCS 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM Press, Oct. 2016, pp. 830–842.