SECURITY ENCLAVE ARCHITECTURE FOR HETEROGENEOUS SECURITY PRIMITIVES FOR SUPPLY-CHAIN ATTACKS

Kshitij Raj
University of Florida
Gainesville, FL, USA
kshitijraj@ufl.edu

Atri Chatterjee
University of Florida
Gainesville, FL, USA
a.chatterjee@ufl.edu

Patanjali SLPSK
University of Florida
Gainesville, FL, USA
atanjal.sristil@ufl.edu

Swarup Bhunia
University of Florida
Gainesville, FL, USA
swarup@ece.ufl.edu

Sandip Ray
University of Florida
Gainesville, FL, USA
sandip@ece.ufl.edu

ABSTRACT

Designing secure architectures for system-on-chip (SoC) platforms is a highly intricate and time-intensive task, often requiring months of development and meticulous verification. Even minor architectural oversights can lead to critical vulnerabilities that undermine the security of the entire chip. In response to this challenge, we introduce **CITADEL**, a modular security framework aimed at streamlining the creation of robust security architectures for SoCs. CITADEL offers a configurable, plug-and-play subsystem composed of custom intellectual property (IP) blocks, enabling the construction of diverse security mechanisms tailored to specific threats. As a concrete demonstration, we instantiate CITADEL to defend against supply-chain threats, illustrating how the framework adapts to one of the most pressing concerns in hardware security. This paper explores the range of obstacles encountered when building a unified security architecture capable of addressing multiple attack vectors and presents CITADEL's strategies for overcoming them. Through several real-world case studies, we showcase the practical implementation of CITADEL and present a thorough evaluation of its impact on silicon area and power consumption across various ASIC technologies. Results indicate that CITADEL introduces only minimal resource overhead, making it a practical solution for enhancing SoC security.

***K*eywords** Security and privacy Hardware attacks and countermeasures · Hardware reverse engineering · Embedded systems security · System on a chip

1 Introduction

With increasing globalization of the design and fabrication processes, the development of a modern microelectronic involves a complex supply chain with a large number of participants dispersed geographically across the world. Most computing devices are developed through System-on-Chip (SoC) designs, which entail the integration of pre-designed hardware blocks (referred to as "intellectual properties" or "IPs") into a single silicon substrate. Players in the SoC design supply chain include IP vendors and suppliers, SoC integration houses, foundries, and testing facilities, Original Equipment Manufacturers (OEMs), and product suppliers, among others. There can be rogue entities in this complex ecosystem that can attempt to subvert the security of the SoC product or operations of other players in the ecosystem. Such subversions can include implanting malicious circuitry or Trojans in the design, reverse-engineering the design functionality, fabricating counterfeit designs, and many others. Obviously, it is crucial to develop security architectures to protect against such attacks.

There has been significant research over the past decade to detect and mitigate supply-chain attacks, resulting in numerous interesting approaches. This includes design obfuscation algorithms, various IC and IP authentication

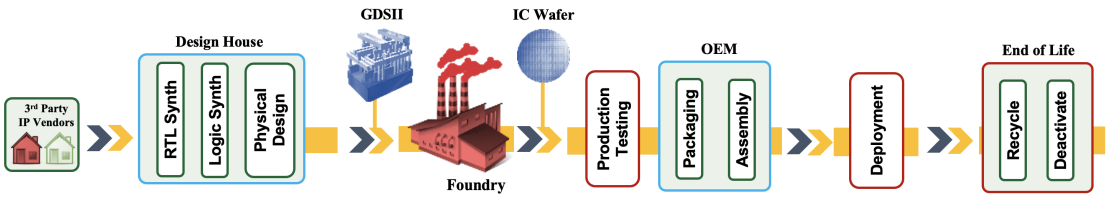

Figure 1: SoC Lifecycle from Supply Chain Perspective. Only major players in the supply chain are indicated.

techniques, design watermarking, trojan detection, and many others. Correspondingly, numerous security engines have also been developed that produce secure enclaves targeting specific threat vectors, *e.g.* AEGIS [1], Sanctum [2] and Komodo [3] provide isolation for trusted execution environments (TEEs), primarily revolving around secure software execution in an untrusted environment. However, — and in spite of its critical need, — we have not found (parameterized) security architectures for systematically integrating heterogeneous solutions into SoC designs or enforcing them at different stages in the design lifecycle for different kinds of security solutions, and no specific architecture targeting an untrusted supply-chain (See Section 7).

In this paper, we develop an architectural framework for the disciplined incorporation of security requirements in modern SoC designs. The key idea is to develop a *skeletal architecture* that can be configured to a variety of security use cases through the integration of custom *security countermeasures (SCMs)*. We exploit this framework to create a systematic, plug-and-play, and configurable architecture, called "CITADEL" for mitigating supply chain threats across the SoC lifecycle. Furthermore, our experimental results demonstrate that CITADEL incurs a very small hardware footprint, making it viable for low-power SoC designs.

The paper makes the following important contributions.

- CITADEL represents to our knowledge the first comprehensive architecture that enables the integration of ad-hoc supply chain security protection in SoC designs through a configurable plug-and-play subsystem. Furthermore, our work represents the first parameterized architectural framework for the systematic integration of SoC security primitives.
- We develop a comprehensive threat model for SoC designs that accounts for different lifecycles, and show how the CITADEL solution can enable systematic protection from the spectrum of threats at each lifecycle.
- We demonstrate the viability of the CITADEL solution through several security case studies as well as detailed overhead analysis on multiple ASIC implementations.

The rest of the paper is organized as follows. Section 2 provides the relevant background on supply chain threats and challenges across the SoC lifecycle. Section 3 gives a high-level overview of our parameterized skeletal architecture and the operating ecosystem. Section 4 describes the CITADEL solution in detail and explains how it incorporates systematic integration of SoC security functionality to mitigate supply-chain threats. In Section 5, we provide detailed use cases demonstrating the versatility of CITADEL. Section 6 provides our experimental results for the evaluation of CITADEL overhead. We discuss related work in Section 7 and conclude in Section 8.

2 Background

Over the past years, microelectronics design has evolved into a globally distributed supply chain incorporating 3PIP (third-party IP) vendors, IC design houses, fabrication plants, and assembly, packaging, and testing facilities. Fig. 1 traces the lifecycle of a microelectronics system across the supply chain. While the globalization of the supply chain has been instrumental in ameliorating the challenges arising from increased design complexities, aggressive time-to-market schedules, etc., an unfortunate side-effect is the growing problem of assurance across the complex supply chain In this section, we briefly recount some categories of supply-chain threats relevant to the work in this paper. Table 1 provides a summary of the threats involved, followed by a brief description.

Table 1: Threat Classification, Mitigation Strategies, and Asset Categories used by CITADEL for Supply-Chain

Class of Threat	Potential Adversaries	Mitigation Approach	Mitigation Technique
Counterfeiting (Piracy and Cloning), Overproduction	Fabrication Facility	Unique and Unclonable Chip Identifier	Unique& Unclonable ChipID obtained using embedded MeLPUF
Reverse Engineering	Testing Facility, Recycling Facility	Design Obfuscation, Redaction	ProtectIP State-Space Obfuscation
Illegal Recycling	Testing Facility, Recycling Facility	Chip Lifecycle Identification and Tracking	Lifecycle validation & tracking with AMI

2.1 Threat Classification

2.1.1 Overproduction and Counterfeiting

An untrusted foundry with access to the design information (layout) may produce counterfeit ICs, including recycled, remarked, and cloned chips that are then sold as legitimate ones to the end user. Note that these overproduced chips, which are then illegally sold in the market, have the same specifications as the original, but the foundry does not have an obligation to meet the functional requirements and carry out extensive validation, as these are sold in the black market or by other interested parties.

2.1.2 Reverse Engineering

Foundries have control over the manufacturing process and can reverse engineer the chips to steal IP design secrets, expose assets, and derive implementation details by deconstructing the die layer by layer. The severity of reverse engineering depends on the lifecycle of the chip. The netlist can be extracted directly from the layout information (GDSII file) at the foundry [4], enabling malicious parties to implant Trojans or alter the functionality of the chip. A malicious testing facility may reverse engineer the chip to steal Chip IDs, cryptographic assets, etc. The situation is further exacerbated by modern-day synthesis techniques, which often leave behind the execution trace of circuit functionality, enabling malicious parties to backtrack the synthesis and steal design specifics.

2.1.3 Illegal Recycling

The global supply chain has seen a sharp rise in the practice and number of recycled chips being re-branded and re-packaged as new and sold, posing a serious threat to commercial and military applications. The harvested chips have inferior quality, security assurance, shorter lifespan, and performance degradation over their lifespan. A 2020 survey done by the Government & Industry Data Exchange Program found a 154% increase in the number of counterfeit ICs traded internationally and a 10-fold increase in seizures of such counterfeit ICs at U.S. borders [5].

2.2 A PUF Implementation

Although the concepts behind the security architecture presented in this paper are general, we focus on instantiating it with a specific architectural instance intended to protect SoC designs against supply-chain attacks (see Section 4). A key security IP used in that instantiation is a PUF module, which creates a unique, unclonable chip ID. The specific PUF module used in our evaluation and experiments is MeLPUF [6], shown in Fig. 2. The PUF exploits the observation that every chip die has a unique doping density and these variations at the sub-micron level result in different capacitance values in the uninitialized bi-stable memory cells. By integrating the cells in specific IPs of an SoC, MeLPUF can generate a unique, unclonable ID for the IP which can act as a watermark; furthermore, by distributing MeLPUF cells across multiple IPs in the SoC, it is possible to collect a signature of the entire SoC that can act as a ChipID. Each MeLPUF bit incurs an overhead of 6 NAND gates.

Realizing such a PUF in an SoC requires, in addition to the distributed MeLPUF cells, a PUF Control Module (PCM) for sending, receiving, storing, and authenticating the PUF challenge-response pairs. Fig. 3 shows a representative implementation of PCM. During the enrollment phase, the PCM provisions the IP IDs, control signals, and the expected responses. IP IDs are first provisioned and stored in the PCM buffer and are in a 1-1 correspondence with all designated IPs. These IDs are then used to index the control signals and expected responses according to their corresponding IP IDs. The PCM transmits the control signal and collects the PUF responses from the IPs. It then compares the signatures with the results of the authentication operation. The PCM also incorporates an error correction module. Error correction

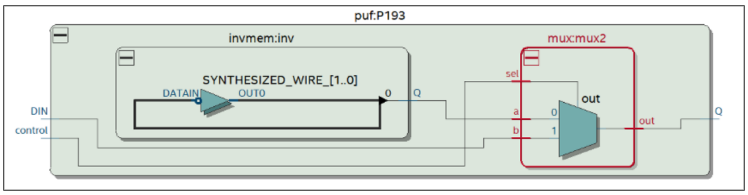

Figure 2: RTL schematic of the MeLPUF architecture generated in Intel Quartus® 18.1. The bi-stable memory element connects to the input of the 2x1 MUX with the circuit data input connected in b. The 2x1 MUX acts as the control element with the control signal being the input to select.

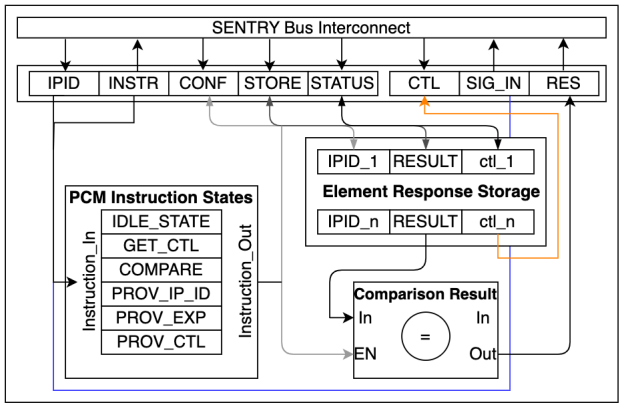

Figure 3: An Overview of the circuitry for the PUF Control Module (PCM).

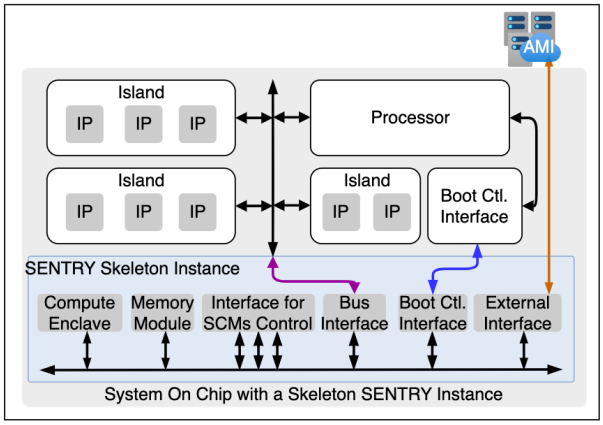

Figure 4: The CITADEL skeletal architecture integrated with a representative SoC model.

is performed on the incoming PUF signatures during authentication. The parity bits for error correction are calculated from the expected signature. The error correction module takes 16 bits at a time, therefore, the PUF signature is split into 16-bit segments for correction. From an architectural perspective, observe that the implementation of the PUF above requires a strategy for integrating individual MeLPUF cells with IPs in the SoC while the PCM can be designed as a standalone IP to perform the coordination and error correction functionality. We exploit this observation in Section 4 by integrating MeLPUF cells with our security wrapper architecture and using the PCM as a standalone IP.

2.3 Design Obfuscation Basics

Hardware Design obfuscation is a technique to protect design IPs from reverse-engineering attacks. Here, an IP is instrumented so that the desired functionality is only realized when a specific input sequence is first applied. The input sequence is often referred to as a key and the application of the key as an initial input sequence is called unlocking. The idea is that the key is not known to the malicious or untrusted party, (foundry), and consequently, the desired functionality is protected from reverse-engineering. Multiple obfuscation schemes have been proposed in two broad categories: (i) combinational obfuscation, where the combinational logic of the design is obfuscated [7], and (ii) sequential obfuscation, where the same applies to the sequential components of the design [8].

3 The CITADEL Skeletal Architecture

The CITADEL architecture is based on a skeletal architectural framework which can be instantiated and configured through individual security countermeasure (SCM) IPs to realize a variety of protection functionalities. The key observation is that the realization of (any) SoC security architecture entails the following ingredients:

- A centralized control unit.
- An interface for secure communication with HOST IPs.
- An interface for secure off-chip communication.

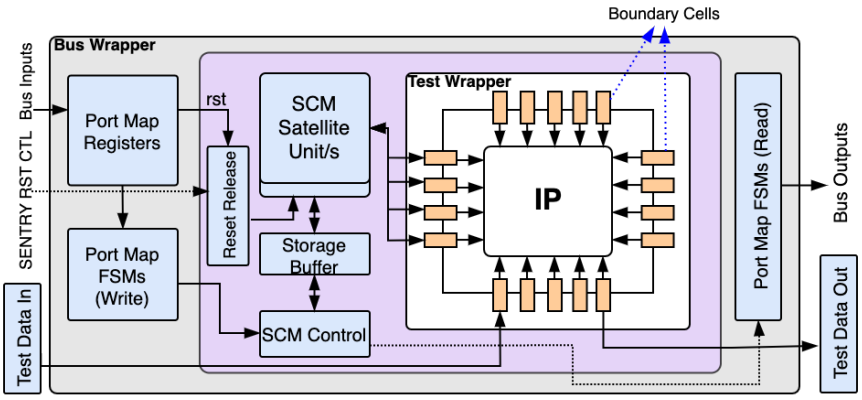

Figure 5: A generic security wrapper with SCM enforcement support.

Each of the ingredients above in fact has specific essential and intuitive functionality. Implementing and enforcing security countermeasures requires a control unit for coordinating and enforcing security countermeasures, hand-offs, data movement across SCMs, and interactions among various entities in the SoC. Any countermeasure involves coordination and communication of the control unit with different IPs, requiring a standardized communication interface. Finally, checking the authenticity of the chip or any hardware IP in the SoC at any point during its operation requires communication with a trusted off-chip entity that keeps track of the transitions of the chip at different lifecycles.

CITADEL incorporates this observation through the skeletal architecture shown in Fig. 4. The control unit is realized through a Compute Enclave (CE) that coordinates and orchestrates actions involved in enforcing security countermeasures. The enclave is implemented with a (lightweight) microprocessor with a standardized interconnect to connect to various SCMs (see below), and its operation and control can be configured through security policies [9] implemented by firmware. The communication of the compute enclave with the IPs in the SoC is implemented through a standardized wrapper, as shown in Fig. 5. The key intuition is that the realization of (any) wrapper to enable communication entails the following:

- An parameterized interface to communicate with the external entity.
- A mechanism to communicate with the internal entity (IP).
- In case of security enforcement, a mechanism to carry out security operations to the underlying IP. This may include accessing specific registers, flip-flops, and other data elements in the IP to extract and/or program specific values at various execution phases.

This generic wrapper enables the enclave to communicate with IPs independent of the underlying IP implementation through dedicated `PORT MAP READ/WRITE` FSMs, which can be customized based on the interface implementation. To facilitate interaction with the underlying IP, the wrappers in CITADEL are implemented as an extension for the IEEE-1500 test wrapper already available with most IPs. The CITADEL compute enclave interacts with the IPs through these communication interfaces (see below). Any transaction t from CITADEL to another IP P in the SoC is interpreted by the wrapper in P, which translates t to operations that need to be performed by P.

While these wrappers are generic, they can be augmented to provide support for security enforcement by integrating SCM satellite units and control, as shown in Fig. 5, to realize security wrappers. The key realization here is that SCMs (may) have a distributed implementation mechanism, which although controlled by a centralized unit, has elements spread across IPs in the SoC. Activating and controlling those elements may require access to specific flip-flops or registers of the IP. The wrappers leverage the fundamental idea that test wrappers have access to low-level registers and flip-flops of design elements at specific instances during execution, using which SCM control and data extraction can be facilitated. This enables control of distributed SCM endpoints by a centralized control unit (CITADEL). The SCMs in the wrapper require two distinct interfaces for communication with CITADEL and the test wrapper to extract or control data from the underlying IP (shown in Fig. 5). As such, they are sandwiched between a bus wrapper, which facilitates communication with CITADEL to receive control packets, and the test wrapper, which performs data extraction and application to the underlying IP for SCM enforcement. The generic wrapper also includes a storage buffer and an SCM control unit to coordinate and manage distributed SCM unit/s in the wrapper, which can be configured through fine-grained security policies. The generic wrapper incorporates a reset release switch, enabling CITADEL to hold IPs

at reset (using the SENTRY RST CTL signal) even when the HOST processor has released the reset signal. Reset gating enables CITADEL to configure and control IPs at boot time to control boot-critical events.

Interface for Secure Off-chip Communication The CITADEL ecosystem includes a cloud-based Asset Management Infrastructure (AMI) for storing and retrieving assets and chip metadata (ChipIDs, obfuscation keys, cryptographic keys, authorization status). CITADEL supports secure communication with AMI through an Ethernet interface.

Remark 1 *One view of the architecture is that the* CITADEL *subsystem acts as a mediator for communication of the IPs with the external world,* e.g.*, AMI. Obviously, an alternate architectural strategy could be for a communication channel to be directly established between IPs (using the IEEE 1500 wrappers) and the AMI. Such an architecture would get away without using* CITADEL *to mediate over this communication. However, a deeper analysis reveals that the communication between the IPs and the AMI needs to account for a* controlled and coordinated *information flow mechanism. Doing so without having a mediator (in this case* CITADEL*) would lead to a complex distributed architecture and require humongous efforts to coordinate any such communication to avoid potential security challenges. In addition to this, the overheads of such an architecture would be significantly more than* CITADEL.

The skeletal architecture discussion above might seem simplistic at first glance. However, there are interesting subtleties that must be addressed to ensure the architecture is indeed viable. An interesting conundrum is the design of the communication interfaces between CITADEL and the HOST SoC. Note that the role of CITADEL is to be in control only of the security functionality of the SoC; other functionalities such as power management or sleep/wakeup needs of the different IPs and communication fabrics are beyond CITADEL's purview[1]. However, the ability of CITADEL to communicate with an IP through the system bus is constrained by whether the bus is active or asleep at a certain point in the system execution. A key challenge is the system boot, during which CITADEL must communicate with IPs to transfer a variety of assets. Obviously, one approach is to simply enable CITADEL to control the power management of the entire SoC. However, doing so would add significant complexity to the CITADEL design and break the principle of using CITADEL as a modular subsystem. Instead, our approach is to include a "boot interface" to enable communication with HOST IPs when the system bus is inactive. Our boot interface implementation enables the communication between the compute enclave and the host processor through a generic GPIO interface, which permits interrupt-based communication with CITADEL.

We conclude this section with an observation of the parameterized nature of the skeletal architecture and the role of security IPs. The skeletal architecture merely provides the *enabling control and coordination framework* for security. The specific protection requirements are enforced through individual security IPs and the software executed on the compute enclave. We discuss one instance of CITADEL below, with specific IPs to protect against various supply-chain threats. Obviously, the effectiveness of any CITADEL instance would ultimately depend both on the efficacy of the individual IP implementations to protect against a specific threat as well as the coordination of that IP with the rest of the SoCs. The key contribution of the skeletal architecture is to provide an architectural framework for enabling such integration: any instance of CITADEL can simply augment the skeletal architecture with individual IPs to create a comprehensive security solution, without the need for "re-inventing" the coordination mechanisms for enforcing the security constraints. With this framework, any such security strategy will be systematically broken into the following steps:

1. Identify the control and coordination functionality with the different IPs in the SoC as well as external interfaces (AMI);
2. Partition the communication and control functionality into software routines and hardware functionality depending on performance and overhead requirements;
3. Develop and integrate the hardware SCMs (if any); and
4. Determine wrapper functionality if the SCM strategy requires communication with other IPs.

4 Instantiating CITADEL Skeleton For Supply-chain Confidentiality Attacks

The skeletal architecture of CITADEL can be instantiated with specific SCMs to enable systematic protection against a variety of threat models. In this section, we exploit this observation to create a CITADEL instance for protection against supply-chain confidentiality attacks.

[1]CITADEL compute enclave is in control of the power management of the CITADEL subsystem, including the internal bus. However, it cannot control the power management of communication fabrics outside the subsystem.

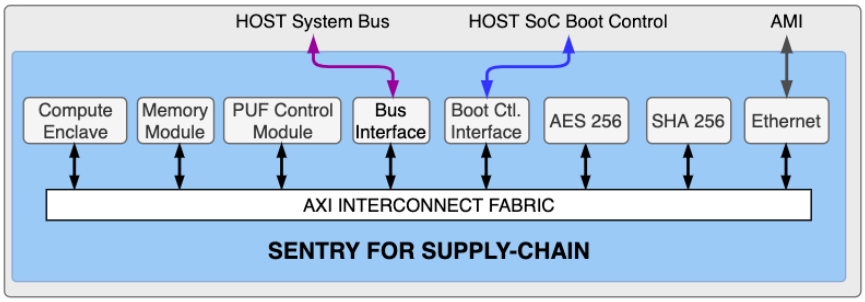

Figure 6: An instantiation of CITADEL skeletal architecture for supply chain confidentiality protection.

Threat Model Our threat model considers untrusted fabrication and testing facilities having access to the fabricated IC as well as rogue users after deployment. The attacks considered include reverse engineering, counterfeiting, overproduction, and illegal recycling. The security requirement is to protect the design features of the SoC against such attacks. Table 1 shows the overall high-level security threats considered. We will show detailed examples of some of the attacks under the threat model in Section 5. Note that the collateral and capabilities available to the different rogue players at different stages of the chip lifecycle, *e.g.*, a fabrication/testing facility where the chip is booted for the first time may have access to the unregistered chip that can be used suppressed from the market and used as a counterfeit while at deployment in-field, the user may only have access to a chip registered with the AMI.

Remark 2 *The threat model for this* CITADEL *instance discussed here is inspired by the DARPA program for assurance of secure silicon (AISS) [10]. The threats correspondingly include supply-chain confidentiality,* e.g.*, reverse-engineering and counterfeiting. However, note that the* CITADEL *architecture itself is agnostic to the specifics of the threat model and consequently permits protection against a diverse class of adversaries through integration of appropriate SCMs. For instance, if the threat classes are extended to include malicious implants, integrity attacks, or fault injection attacks, the corresponding* CITADEL *instantiation would "merely" require an SCM for detecting/mitigating Trojan actions or fault injection incorporated as a security IP. Indeed, in previous work, Trojan detection was considered through a similar centralized subsystem by integrating fine-grained security policies [11].*

Given the above threat model, our high-level security strategy is to use an unclonable physical signature of the chip as a chip ID to protect against IC counterfeiting by an untrusted foundry or testing facility. Correspondingly, hardware design obfuscation is used to address reverse-engineering attacks. In our implementation, we use the unique signatures obtained from MeLPUF and the obfuscated designs using the ProtectIP [7] framework. However, when these methods are applied in an SoC, there are architectural issues that need to be addressed. Following are some of the representative issues.[2]

- How will the PUF signatures be generated and communicated?
- How will the design obfuscation keys be provisioned by CITADEL? How can we make sure they are inaccessible to untrusted players?
- How will the coordination among IPs be implemented for creating ChipID through PUF signatures? How will the signatures be extracted from individual IPs?

CITADEL enables the architect to systematically analyze the questions above and develop disciplined architectural solutions to address them.

4.1 Hardware Instantiation of the CITADEL Skeleton Architecture

Security Countermeasures (SCMs) We use SCMs to implement (1) the necessary secure storage (*e.g.*, for obfuscation keys and chip ID created from MeLPUF), (2) the PUF Control Module necessary for MeLPUF and the MeLPUF wrapper as the SCM satellite unit, and (3) the key application SCM satellite unit necessary for obfuscation, as shown in Fig. 7.

[2]Another challenge is to protect the sensitive communication of a newly fabricated IC, possibly in an untrusted foundry or testing facility with AMI. Such communication is required at first power-on, *e.g.*, to enroll with AMI the chip ID collected from the PUF signature. For the CITADEL instance discussed here, we assume that these communications are performed through the mediation of a trusted hardware security module (HSM). This is consistent with current industry standards [12][13].

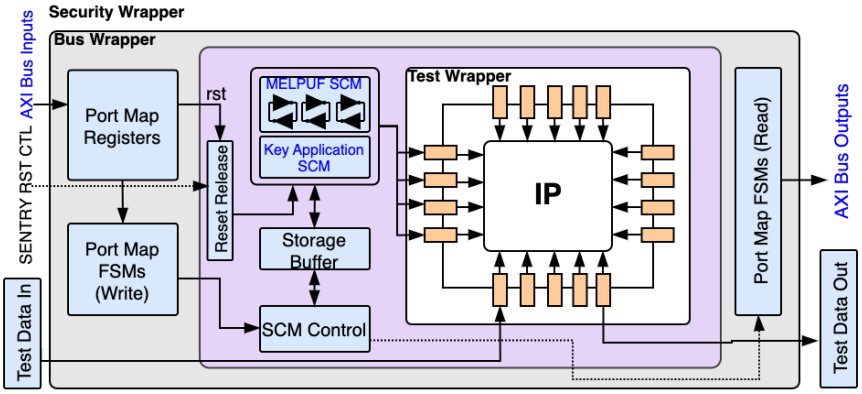

Figure 7: A security wrapper (augmented generic wrapper) with integrated MeLPUF and key application SCM on a HOST IP.

Remark 3 *SCMs (MeLPUF [6] and ProtectIP [7]) are integrated with the* CITADEL *instance since they are representative technologies. Security protections in today's practice is an arms race between attack and defense, so one can envision that these technologies will be broken in the future, and likely new innovations will come in protection mechanisms superseding these technologies. If these technologies are broken, obviously the security provided by the* CITADEL *instance incorporating them will be broken too. However,* CITADEL *itself is an architectural framework independent of the specifics of the protection technology. We do not take a position specifically defending these technologies. Rather, the goal of the architecture is to enable disciplined integration of* diverse *protection mechanisms in SoC without requiring hand-crafted custom security implementation.*

Enclave Software We implement the compute enclave using the Bluespec RISC-V processor core. The software running on the compute enclave is responsible for control and coordination of SCMS (MeLPUF and design obfuscation, a.k.a the PUF and logic locking functionality), *e.g.,* it initiates unlock requests for locked IPs during boot and initiates commands for extraction of PUF signatures from IPs during the chip enrollment process. It also mediates the coordination between CITADEL and the host processor during system boot, manages communication with the AMI for SoC and asset enrollment, and controls the SoC throughout its device lifecycle.

Security Wrappers These are generic wrappers augmented with various two SCMs needed for enforcement of security primitives by CITADEL. They are responsible for extracting and application of control information from individual IPs. The functionality of the wrappers are extraction of MeLPUF signatures (SCM1) and the application of obfuscation keys (SCM2) under the command of the enclave software. Fig. 7 shows the wrapper design with MeLPUF and key application functionality included. Note that the functionality is positioned amid the IEEE 1500 test wrapper and standardized AXI-4 bus wrapper.

Extensibility and Modularity As explained above, a key aspect of CITADEL is its modularity and flexibility to account for various aspects of the threat model as well as the overhead constraints. As an obvious example, the Bluespec RISC-V core can be swapped with a custom RISC-V core to meet the design requirements. A more subtle aspect is the extensibility to handle refinement and sophistication of the threat model. For instance, the threat model discussed above can be viewed as a "base level" threat model in the sense that it did not account for possible malicious access to the chip ID or obfuscation keys either from the key storage module or during communication. Accounting for that simply requires extending the enclave software with security policies for access control [14], and appropriately adjusting the IPs to account for adversary capabilities, *e.g.*, if the adversary is assumed to have side channel access to stored data then the storage IP would be extended to a "vault", *i.e.*, tamper-proof storage that is resistant to access via side-channel [15] [16]. Correspondingly, eavesdropping threats during communication can be addressed through security policies controlling encryption, which in turn extends the security IPs with crypto modules (*e.g.*, AES and SHA modules), which are already available in this instance of CITADEL. Indeed, our current implementation, shown in Fig. 6 accounts for an extended threat model that includes adversaries with side channel access power and the ability to eavesdrop on communications.

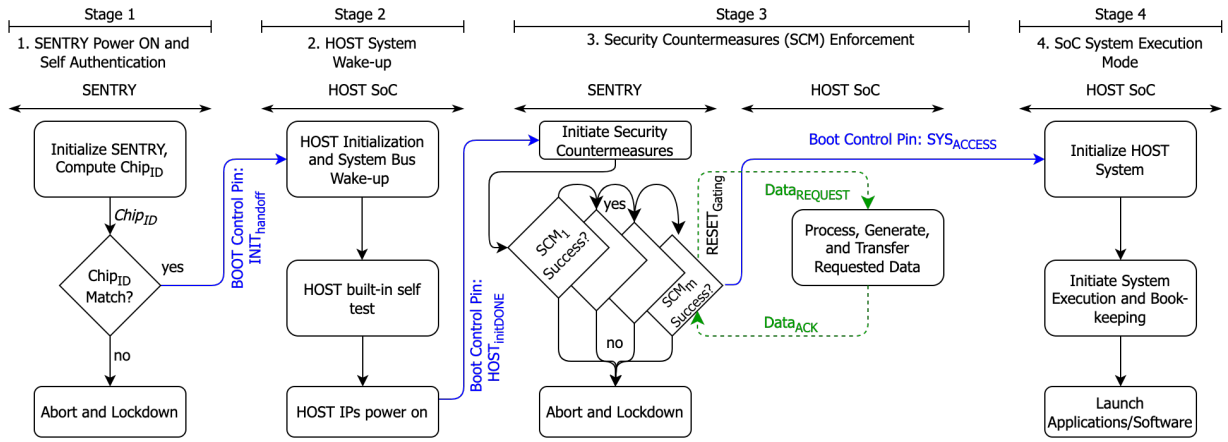

Figure 8: Boot interaction flow diagram between CITADEL and HOST SoC via the Boot Control Interface for a Supply-chain use-case. Only communication between CITADEL and HOST is shown.

4.2 CITADEL Software Control

The CITADEL skeletal architecture provides a generic hardware architecture, which enables the specific control and coordination functionality to be moved into a microcontroller firmware implementation, which is different from any boot firmware to be executed on the main processor of the SoC. The architecture provides hardware-level hooks to simplify the firmware executed in the compute enclave. Such layering results in flexible instantiations of CITADEL, as the same skeletal architecture can then be deployed for various security mechanisms. For this instance of CITADEL for supply-chain, the software control consists of a 4-stage sequence between itself and the HOST SoC, as shown in Fig. 8. This firmware design standardizes the interaction between CITADEL and the HOST, which demarcates their responsibility during system boot. While CITADEL is responsible for *only* the security aspect, the HOST is responsible for the SoC wake-up and boot. As a result, the size and complexity of the security firmware of CITADEL reduces significantly, eliminating the need for bootstrapping in the case of CITADEL's boot sequence. – However, CITADEL controls of initiation of the HOST system boot after successfully booting up its security subsystem, as shown in Fig. 8.

CITADEL Boot Control Sequence The complexity of (any) boot protocol arises from three main factors: (1) self-stabilization, (2) secure management of assets and information flow, and (3) bootstrapping. In the case of supply-chain, self-stabilization includes lifecycle attestation, remote authentication with AMI, etc. Similarly, asset flow includes the transfer of PUF responses from HOST IPs to CITADEL, application of obfuscation vectors to IPs, communication with the AMI, etc. As highlighted above, bootstrapping does not apply to this instance of CITADEL for supply-chain. In particular, CITADEL provides architectural features to close this gap and simplify the overall boot process by decoupling the actions of the host CPU to wake up different system components from the security-related actions, such as unlocking and authentication of the constituent IPs. A key requirement in enforcing SCMs at boot is to enable a secure transfer of data (assets including PUF signatures, watermarks, cryptographic keys, etc.) from the HOST to CITADEL. An interesting conundrum for boot is the retrieval of data from the HOST to CITADEL. Since the main system bus is inactive in the first phase of boot, CITADEL issues an interrupt using the *Boot Control Interface* to the HOST for initialization and system bus wake-up. This is followed by an acknowledgment from the HOST to CITADEL indicating successful Phase 2 operations. At this stage of the boot, the system bus is active and SCMs can be enforced. SCMs can be executed in chaining or independent of preceding SCMs, contingent upon the nature of the SCMs or the devised mitigation strategy. To facilitate data retrieval from the HOST for SCM execution, each HOST IP can be triggered in isolation to retrieve the data required, while other IPs are held at reset. This protection also ensures that the data in transmission is only visible to the source IP and CITADEL. SCM executions can be configured through firmware or security policies, allowing for modularity and extensibility in the software stack as well. After all successful SCM executions, CITADEL relinquishes system access to HOST via an interrupt, after which the SoC can initiate system operation and launch applications.

Remark 4 *Note that in this instance of* CITADEL *for supply-chain, the boot control sequence accounts for hardware-based attestation, while the responsibility for system firmware authentication is delegated to the HOST processor. This*

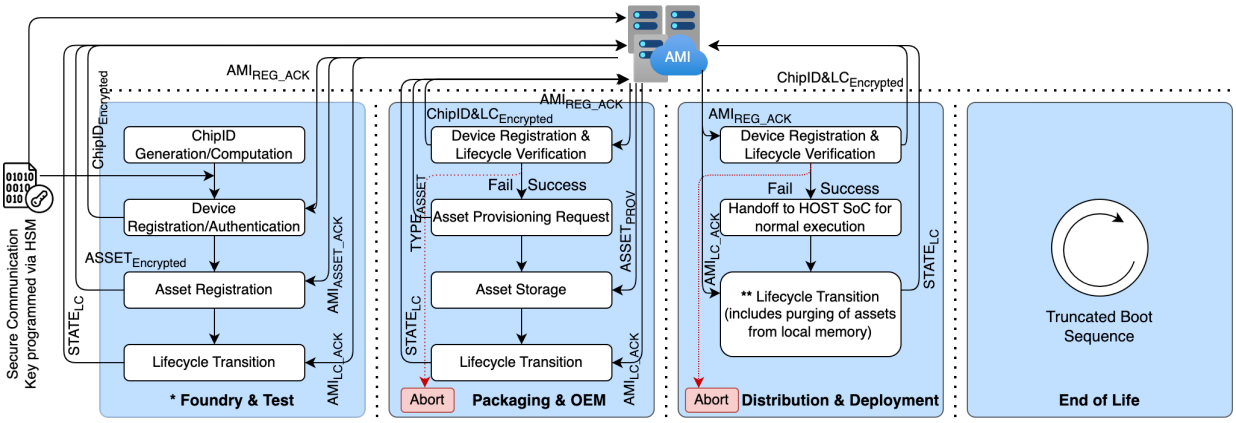

Figure 9: Lifecycle management using CITADEL's boot control sequence. Only communication with the AMI is shown.
* The SoC is under the control of the Hardware Security Module (HSM) during chip birth and registration only.
** It should be noted that lifecycle transitions from deployment to recall are under the control of the OEM. Hence, it is crucial to purge any end-user assets, firmware, and application code from the device and update the corresponding status of the device with the AMI to avoid unauthorized asset requests from the AMI.

is solely due to the definition of threats laid out by the threat model under consideration. It is certainly possible to migrate this functionality to CITADEL*, as has been demonstrated in other academic work [17].*

Some interesting subtleties arise in the boot control sequence for an untrusted supply-chain environment when accounting for the different device lifecycles (Fig. 1). As the device transitions from one lifecycle to the next, the execution environment (including threats) and ownership of the SoC change, and consequently, the CITADEL's boot control needs to account for such subtle alterations. For example, after registration of the device with the AMI at the testing lifecycle, subsequent boot stages in the OEM lifecycle no longer involve device registration; instead, they account for device attestation and verification of its registration status, as shown in Fig. 9. As such, some activities that need to be accounted for include:

- Registration (and attestation) of system hardware integrity.
- Secure provisioning of identities (IDs) from AMI.
- Mutual authentication of components within a system or a package.
- Supply chain tracking, including device disenrollment.

4.3 Device Lifecycle Support

Any microelectronics device transitions through a sequence of lifecycles. By a "lifecycle", we mean the duration of time when the device is in the custody of a specific stakeholder, *e.g.*, foundry, Original Equipment Manufacturer (OEM), user, etc. Obviously, transitioning from one lifecycle to another impacts the targeted security profile, *e.g.*, the degree of access of an OEM is different from that of the user. CITADEL provides explicit architectural support for lifecycle transitions, including the following activities.

- Secure provisioning of identities (IDs) and/or firmware.
- Attestation of system hardware authenticity and integrity.
- Update of IDs.
- Mutual authentication of components within a system or a package.
- Supply chain tracking including device dis-enrollment, and optionally re-enrollment.

Fig. 10 shows lifecycles supported by CITADEL. The idea is to consider a chip being "born" (powered on for the first time) in a possibly untrusted testing facility and then sequence through "Packaging", "Deployment", "Recall", and "End of Life". As with any other architectural features, the sequence can be configured: one lifecycle stage can

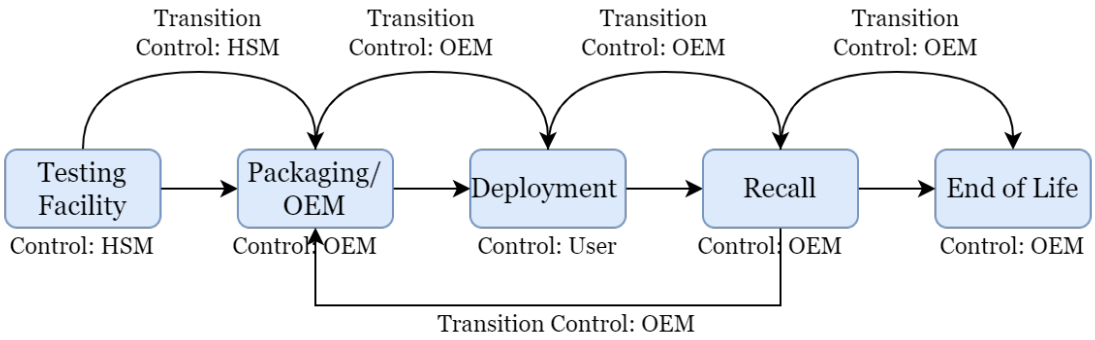

Figure 10: SoC Lifecycle Stages Supported by CITADEL. *Control* represents the entity having physical access and control of the chip in the respective lifecycle and *Transition Control* indicates the entity authorized to transition the lifecycle state of the device from the current state to the next.

be split into multiple more fine-grained stages (*e.g.*, the "Deployment" stage can be split to comprehend the security operations when the device passes from the custody of one user to the next), and two stages can be fused (*e.g.*, "Recall" and "End of Life" can be fused if the device is disallowed to pass back into the open market after recall). Nevertheless, these five stages are representative of real-life microelectronic lifecycles. Each lifecycle has its own set of security and non-security operations, data (including assets) generation, movement, and processing as discussed below.

Manufacture and Test: This lifecycle represents the steps between fabrication to acceptance when chips generate their first set of unique identifiers (ChipID). During this lifecycle, the chip is under the custody of a possibly untrusted testing facitlity, and manufacturing tests are done to the design.[3]

Packaging and OEM: SoCs are then integrated after fabrication/test to a system board or system-in-package. This includes the ability to store additional assets (firmware signature, scan keys, etc.). Furthermore, a security process for mutual authentication between the SoC and the system integrator is also invoked in this lifecycle as a part of CITADEL's security policies.

Deployment: This involves the distribution of packaged systems to end users. CITADEL performs self-authentication at power on at first boot in this lifecycle and upon successful authentication, releases the HOST processor to enter normal execution mode.

Recall: The core idea is to ensure that the only long-term end-user changes to the chip or device are (1) physical changes due to use and (2) manufacturer or integrator-approved upgrades. At this stage, the OEM can decide on whether to re-enroll the device or decommission the device entirely. Some use cases may allow a device to re-enter the supply chain after dis-enrollment, for example through authorized device recycling or refurbishment channels. In these cases, steps for the `OEM` lifecycle for re-enrollment are repeated. After a device is dis-enrolled, it could also be disposed of. CITADEL ensures that all data, whether it is from the manufacturer, integrator, or OEM is removed from the device, and the AMI is also updated to reflect this new state of dis-enrollment. It is important to note that this transition might be skipped entirely. For example, a laptop could be run over by a bus which means it is effectively disposed of despite not going through the transitions.

End-of-Life: In this lifecycle, no operations are allowed to happen, and the SoC enters a truncated boot sequence. Once the OEM transitions the device lifecycle to `End-of-Life`, all assets, including the ChipID, communication keys, UUIDs, etc. are erased from CITADEL, except the lifecycle state of the device. Correspondingly, the status of the chip is also updated with the AMI as soon as the device transitions to this lifecycle, and this happens at `Recall`.

[3]These tests are generally not functional tests and involve verifying basic read/write and register access tests, stuck-at faults, X propagation, power-on tests, etc. to verify fabrication correctness to an extent.

A key activity on the first power on is the generation and registration of ChipID.[4]. After successful registration of the ChipID, other assets such as the lifecycle state, PUF CRPs, obfuscation vectors, communication keys, and other relevant assets are registered or provisioned from the AMI. This process is usually done using an HSM and the ATE (Automated Testing Equipment) using which some of the security-critical assets are registered and provisioned. ChipID and other assets in this lifecycle require the assistance of an HSM to mediate the interaction between the AMI and CITADEL. Upon successful completion of the above-mentioned processes, the HSM transitions the lifecycle state from `FABRICATION/TEST` to `PACKAGING/OEM`.

Remark 5 *Direct communication with the AMI is required at the first boot after chip-birth during the Tesing lifecycle, as shown in Fig 9. During this lifecycle,* CITADEL *provisions an asset called "lifecycle validation key", which is used to authenticate the AMI and entities attempting to transition the lifecycle of the device. This key is provided by the OEM to the HSM. Lifecycle transition is a security functionality as different lifecycles involve distinct functions (in the form of security policies, which include authentication checks, IP unlocking, lifecycle transitions, etc.) and operations in the SoC, hence requiring a gatekeeper to authenticate such transitions. Each lifecycle transition has a unique validation key, which needs to be provided while attempting to make such transitions. The instance of* CITADEL *for supply chain in this paper uses 256-bit lifecycle validation keys for each lifecycle.*

5 Supply-Chain Security Case-studies With CITADEL

CITADEL has been used to develop security solutions corresponding to a variety of supply chain threats. In this section, we discuss several representative use cases. The case studies shown in this section are representative, but not comprehensive. We discuss them in detail to provide a flavor of different threat mitigation designs using CITADEL.

5.1 Threat Class I: SoC/IP Counterfeiting and Overproduction at Chip Birth

Threat Description

Consider a (possibly untrusted) fabrication and testing facility where the chip is powered on for the first time. The threat entails the untrusted testing facility snooping ChipID during the registration, and using the snooped value to register a counterfeit chip. At this stage, the chip is born and has not yet been registered with a central facility, and the ChipID is a critical asset that needs to be safeguarded.

Security Strategy The security mitigation makes use of the available PUF functionality in CITADEL. The security architecture integrates PUF triggers on a (user-defined) subset of N IPs in the SoC, whose signatures are used to generate a unique and unclonable ChipID, which is registered with the AMI. Finally, to avoid the possibility of ChipID and other IDs being snooped by other IPs while they are being transferred over to CITADEL via the system bus, the security wrappers use the reset gating logic controlled and orchestrated by CITADEL. Note that the HSM is in control of the SoC throughout the security enforcement process in this scenario.

Implementation Fig. 11(a) shows the flow of operations for implementing the strategy above. It is divided into three phases outlined below.

- Chip ID Computation: During the chip birth boot-up of the chip, the chip establishes a connection with the AMI using a Hardware Security Module (HSM). The chip generates PUF golden response pairs from N IPs and computes the ChipID, which, along with the lifecycle state is programmed onto the chip via the HSM. The ChipID along with the lifecycle state is encrypted using the secure communication key provided by the HSM and transferred to the AMI.
- AMI Registration Check and Enrollment: If AMI realizes that the chip is already registered, the process is terminated, and a response is sent back to CITADEL stating that the registration was unsuccessful. If the chip operates in an illegal lifecycle, it re-boots in the previous lifecycle. If the ChipID registration is successful, the lifecycle record of the chip is updated in the AMI ledger, indicating that the chip has transitioned from the initial registration process, and subsequent registration requests from this chip will not be served.
- Asset Enrollment: After completion of the chip registration, CITADEL proceeds to register IPIDs (hashed PUF responses) and the obfuscation vectors (using the HSM) with the AMI. During the entire sequence of operations, the chip is to be physically placed inside the HSM. Assets and data transferred from CITADEL to the AMI are encrypted with a "secure communication key", which is generated by the HSM and shared between both parties (chip and the AMI).

Remark 6 *The rationale behind using the HSM in an untrusted environment (foundry and test in this case) relies on the notion that the HSM facilitates a small fraction of communication with the AMI which can be secured independently. This has been a topic of active and significant research [13][18][12]. They establish how an HSM can work securely in an adversarial environment, where the key idea is modeling the HSM like an ATM, where while the HSM can be potentially physically destroyed, its actions and communication cannot be corrupted.* CITADEL *leverages this as a baseline assumption for securing communication with the AMI at chip-birth and receives a* `communication key` *from the HSM, using which any and all data or assets are communicated off-chip throughout the remainder of the device's operational life.*

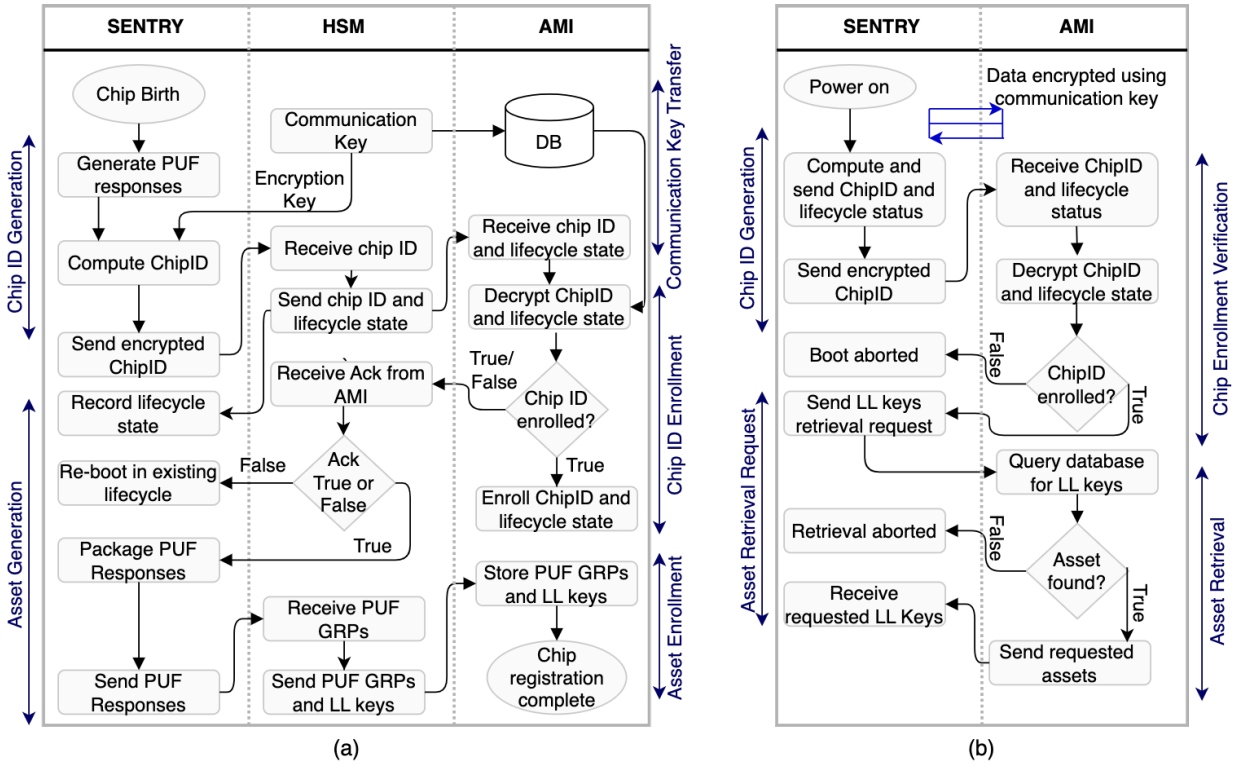

Figure 11: (a) ChipID registration and asset enrollment flow (b) AMI authentication & asset provisioning flow

To ensure the integrity of the assets being transferred off-chip during communication with the AMI, CITADEL *protects them using encryption and hash functions using the AES-256 and SHA-256 IPs inside its secure boundary. The ChipID is computed by XORing all values of PUF responses from all MeLPUF instances in the HOST SoC and hashing them to get a 256-bit unique ID (ChipID =* $f_{SHA\text{-}256}(PUF_0 \oplus PUF_1 \oplus ...PUF_i \oplus)$*).* CITADEL *also stores the individual PUF responses and the obfuscation vectors using the same function. During communication with the AMI, each asset is encrypted using the* `communication key`*, using the function* $OA = f_{AES\text{-}256}(Asset)$*, where* OA *is any outgoing asset to the AMI.*

5.2 Threat Class II: Reverse Engineering

Threat Description

Consider a threat where an untrusted foundry, testing, or decommissioning facility (all possibly untrusted) attempts to reverse engineer the SoC (or parts of it) to extract privileged information regarding the design, IP secrets, and make alterations to the system functionality.

Security Strategy The mitigation strategy involves the design obfuscation of unsanitized zones in the SoC. Unsanitized zones are zones that need to be protected against reverse engineering attacks. This is done using the ProtectIP obfuscation scheme [7] technology by obfuscating parts of the design (unsanitized zones).

Implementation CITADEL uses an obfuscated netlist of the IP, whose design is functionally equivalent to the original design. Fig. 12 shows the modified netlist of the design where only one state path unlocks the design into its functional obfuscated space through the application of a series of unlocking keys. In this implementation, N IPs in the SoC are protected against reverse-engineering attacks. The unlocking procedure is divided into two phases outlined below.

- Asset Provisioning: The logic locking keys for the obfuscated IPs are provisioned by CITADEL upon successful registration & authentication of the chip with the AMI post-testing, as shown in Fig. 11(b).

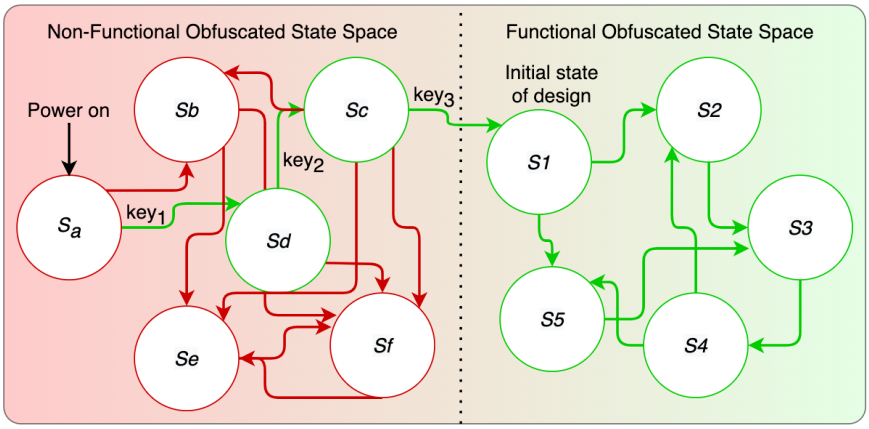

Figure 12: A modified state-transition diagram for an obfuscated design with additional transition states.

Each key is stored on the secure on-chip memory inside CITADEL with a unique identifier indicating the corresponding IP.

- Key Application for Unlocking: The logic locking keys are transferred to the security wrapper of obfuscated IP instances during system boot and it programs the keys/activation package, traversing through the obfuscated FSM ($Sa \rightarrow Sd \rightarrow Sc \rightarrow S1$), as shown in Fig. 12, and the IP is unlocked. Obfuscated regions can similarly be locked back to their original state by CITADEL if specific violations are detected or security policies are violated.

5.3 Threat Class III: Illegal Recycling at an Untrusted Facility

Threat Description: Consider a case where a chip is sent to a recycling facility to be decommissioned and recycled for components or materials. The facility (possibly untrusted) may attempt to re-brand the chip with new markings and physical identification and sell it in the open market for financial gains. While preventing physical alterations to the chip is beyond the scope, the management and control over the activation and enrollment of decommissioned chips present a realistic approach to protect against such attacks.

Security Strategy: CITADEL maintains the chip's registration and lifecycle status with the AMI using which the chip can be tracked throughout the lifespan of the chip and control re-enrollment back into the supply chain. The lifecycle status is modifiable under specific circumstances by *authorized* entities (see Fig. 10), and once a chip transitions to the decommissioning lifecycle, the AMI marks the chip as *deactivated* or *decommissioned*, upon which, it cannot be re-enrolled into the supply chain. CITADEL also maintains its current lifecycle value locally using which it boots on power-on. Lifecycle verification is the first check done during the first boot at power-on in the respective lifecycles. In the case where it does, the lifecycle status and the AMI record ensure that the chip falls into a truncated boot sequence, rendering it inoperable.

Implementation: The AMI keeps a record of the chip metadata to track the illegal recycling of the chip. When the chip transitions from OEM to deployment or decommissioning, the metadata (ChipID, current lifecycle) of the chip is updated accordingly in the AMI ledger. Upon first power on in each lifecycle, the chip undergoes a truncated boot sequence to validate the existing lifecycle after successful validation of ChipID. This sequence of steps is performed on every chip lifecycle transition before the chip is booted up in the new lifecycle to ensure the boot-up, hand-swap or further actions are legal. If the validation process fails, CITADEL boots the chip in the preceding lifecycle. Now, if a malicious recycling facility repackages the chip and attempts to sell it in the open market, the chip will not be able to successfully boot up as the current lifecycle status of the chip indicates the chip has transitioned to decommissioning.

6 Overhead Evaluation of CITADEL Instance for Supply-chain

Integrating a security engine solution involves overhead in various parameters, including power, area overheads, and timing delays. In this section, we perform an empirical evaluation of the overhead of the CITADEL instance for supply-chain security. To our knowledge, there is no other comprehensive SoC security architecture for supply-chain

attacks for systematically implementing ad-hoc supply-chain security solutions to enable direct comparison with CITADEL. However, our evaluation of CITADEL demonstrates that it has a minimal area footprint and incurs a very small cost in all the relevant overhead parameters.

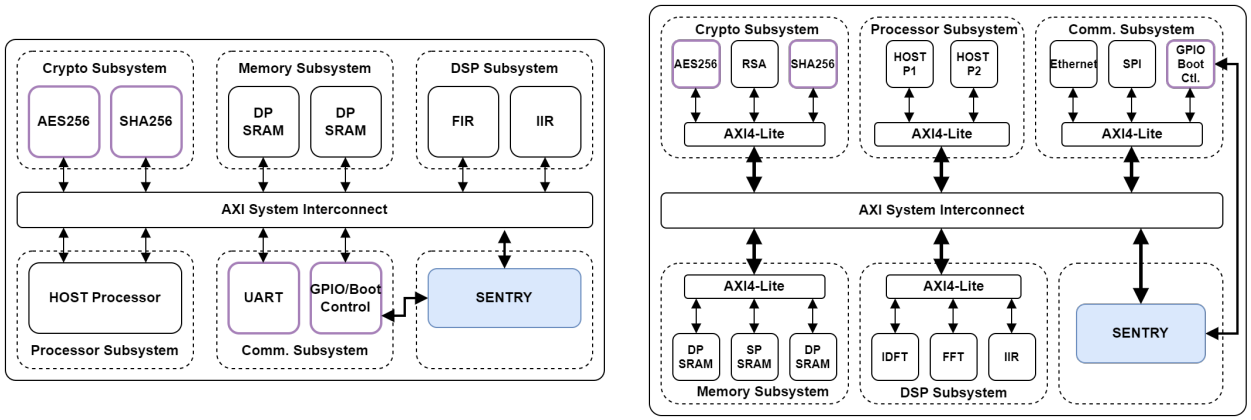

Figure 13: SoC models used for CITADEL evaluation. (a) Single-bus SoC: This SoC model has a single shared-bus architecture. (b) Multi-bus SoC: This SoC model has an AXI4 interconnect as the primary bus and five AXI4-Lite buses as subsystem interconnect. Five AXI4-Lite bus-based peripherals/hardware accelerator subsystems are connected to the AXI4 bus through standard bus bridges. Both designs have CITADEL integrated.

To address the dearth of available open-source SoCs for evaluation of CITADEL, we developed two SoCs, Single-bus and Multi-bus SoC architectures inspired by industry designs shown in Fig. 13 and integrated CITADEL into the SoC testbeds.[5]

The Single-bus SoC is an area and power-efficient implementation of an SoC for IoT devices targeting low-power applications. The Multi-bus SoC is significantly more complex than Single-bus and represents a tiled-hierarchical architecture with application-specific subsystems. Both SoCs are functional SoC implementations developed to verify the efficacy and cost evaluation of CITADEL. CITADEL was also integrated with the MIT CEP SoC [20]. All these designs were synthesized on two technology nodes (LEDA 250nm and GSCL 45nm), and the overhead ratios of SENTRY with the mentioned SoC architectures are provided in Table 3. Standalone CITADEL area and power metrics are provided in Table 2.

The augmentation includes two layers of security primitives: (1) design obfuscation using the ProtectIP obfuscation scheme, and (2) 256-bit MeLPUF instances have also been inserted for the construction of ChipID. These instances are obfuscated with a 512-bit vector. For unlocking, these vectors are sequentially applied over P clock cycles. Each keyframe is fragmented based on the width of the input signal width for the underlying IP (excluding clock and reset), and the number of iterations is calculated.

6.1 CITADEL Area Overhead

We synthesized SoCs with CITADEL integrated onto two ASIC libraries, (1) LEDA (250nm) and (2) GSCL (45nm) using Synopsys Design Compiler T-2022.03-SP5, and a comparative analysis is provided in Table 3. The major contributing components to area and power overheads were the Bluespec RISC-V core, AXI interconnect, and the 64KB system memory. The security wrappers augmented onto the IPs in the SoC in Fig. 13 (indicated with the purple outline) also contribute to the area overheads of CITADEL although they are beyond the CITADEL boundary. This is because the security primitives implemented by CITADEL depend on the security wrappers to enforce those security primitives. The overheads from the 256-bit MeLPUF instances in the testbed SoC are also counted because they are used to generate the ChipID of the SoC. This overhead estimation of the ChipID asset is done by evaluating the number of gates required for each bit of the asset. As such, the area overhead in terms of NAND-2 equivalent gates can be

[5]The SoCs are developed through an automated CAD flow [19]. We have extended the flow to generate CITADEL instances for security. The flow can currently generate bus-based SoCs and supports single clock and reset domains. Note that CITADEL instance is a standalone subsystem, similar to IPs in the SoC, thus simplifying its integration process into SoCs like any IP block, provided the interface requirements for communication with the HOST processor are satisfied.

Table 2: Area & Power Metrics of CITADEL

Technology	Die Area (um2)	Dyn. Power (mW)	Leak. Power (mW)
LEDA 250 nm	12783906	551	15
GSCL45 45 nm	756716	84.2	3.5

calculated using the expression:

$$Overhead_{PUF} = \sum_{0}^{N} \{ChipID_{bits} \times 6\} \tag{1}$$

The overall overhead of CITADEL with all its distributed SCM controls can be calculated using the expression:

$$Overhead_{Total} = Overhead_{SENTRY} + \sum_{0}^{N} Sec.Wrapper_{um^2} + Overhead_{PUF} \tag{2}$$

For a standardized CITADEL architecture for supply-chain, we concluded that such overheads (2) in terms of both power and area would be minimal, as shown in Table 2.

Table 3: Area & Power Overhead of CITADEL integrated with SoC Testbeds

SoC	Baseline Statistics		with SENTRY		Overheads	
	Area (um2) LEDA	Area (um2) GSCL	Area (um2) LEDA	Area (um2) GSCL	Average Area (%)	Average Power (%)
Single-SoC	82281134	3943861	95065040	4700577	17.50	17.89
Multi-SoC	106245988	4561205	119029894	5317921	14.00	12.45
MIT CEP	141211642	5887352	153995548	6644068	10.50	10.91

6.2 Security Wrapper Area Overhead

Security primitives enforced by CITADEL require IPs to be augmented with the security wrapper. Since in a typical SoC, all IPs are not secured, we compute the overheads using representative IPs. To obtain the representative overhead values for the security wrappers, each wrapper instance was synthesized for the GlobalFoundries GL12LPP 12nm technology. Table 4 shows the results. Note that the overheads are minimal (*e.g.*, 0.70% area overhead for AES-256) when compared to the overall footprint of the underlying IP. The security wrapper overhead remains approximately constant in terms of the gate count and only varies due to the variation in the size of the buffers and MUXs implemented to switch data based on different addresses for read and write operations, based on the memory map of the IP.

Table 4: Area & Power Overhead of Security Wrapper Instances From Synthesis on GlobalFoundries GL12LPP (12nm)

IP	Baseline Statistics		with Security Wrapper		Overheads		
	Area (um2)	Gate Count (K)	Area (um2)	Gate Count (K)	Area (%)	Gate Count (%)	Total Power (%)
AES-256	58628.10	404.00	59049.44	406.90	0.70	0.70	0.91
SHA-256	2691.80	18.55	2800.62	19.30	4.04	4.04	1.82
UART	2236.30	15.41	2323.36	16.01	3.89	3.89	2.46
GPIO	1352.50	9.32	1423.2	9.87	5.90	5.90	2.85

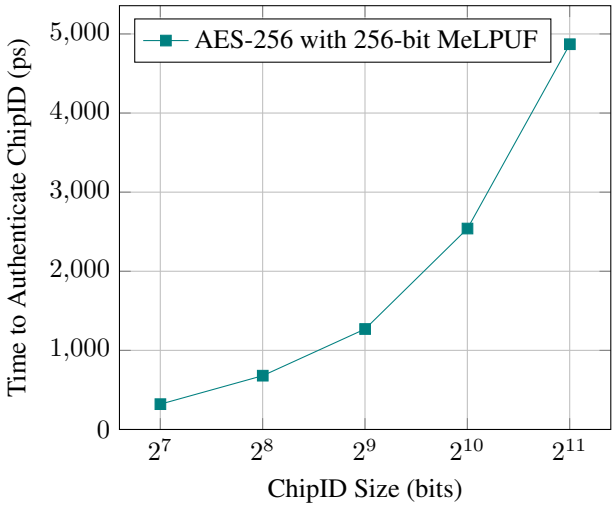

Figure 14: ChipID size vs authentication delay chart.

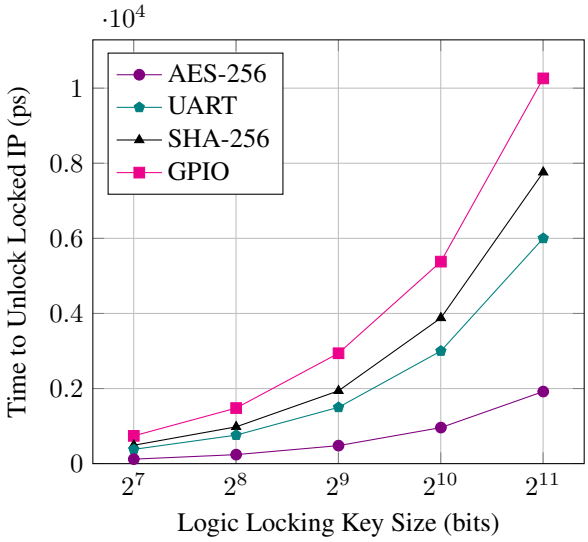

Figure 15: IP unlocking key size vs unlocking delay chart.

6.3 Evaluation of Security Operations

To determine the timing delays incurred from security operations, we analyze (1) ChipID authentication, and (2) Obfuscated design unlocking as a function of the ChipID size and the obfuscation key vector size, respectively. Fig. 14 shows the variation in time taken to complete the ChipID authentication sequence with different ChipID sizes. Note that the range of widely used ChipID size ranges from 128 bits to 512 bits. In this range, the delay incurred due to the authentication operation is negligible. Fig. 15 shows the variation in time delay when the unlocking key size is increased. We analyze this delay using the UART and AES-256 instances in the HOST SoC. The difference between the UART and AES-256 for the same key sizes is from the width of input signals to each instance. In this case, the width of the input signals of AES-256 is greater than UART, thus, larger segments of keys can be applied to the IP FSM at once, resulting in reduced unlocking delay. The feasible key sizes for unlocking are estimated to be from 192 bits to 512 bits, and in this range, the delay incurred due to this operation is minimal.

7 Related Work

There has been significant research on developing systematic and streamlined frameworks for security architecture and the implementation of modern SoC designs. We categorize the related work in two broad categories: *(1) Centralized Platforms and Standards*, as shown in Table 5, and *(2) Heterogeneous and Isolated Solutions.*

7.1 Centralized Security Platforms and Standards

Security frameworks have been designed to enable isolation of functionality at different levels of trustworthiness through Trusted Execution Environments (TEEs). One well-known TEE architecture is the Trusted Platform Module (TPM) [21]. TEE frameworks like Apple Secure Enclave [22][6], ARM TrustZone [24], and Intel SGX [25] were developed to provide segregation of the area of memory and CPU that is protected from the rest of the CPU using encryption. Security enclaves like Apple's and similarly others can be re-targeted for supply chain security, — however, re-purposing would include significant complexity and changes in architecture. Doing so without innate architecture support generally implies a high software and design complexity.

RealSE [26] presents another security engine candidate, built around providing security for crypto-based processors and devices. Several other works [27] [28] have also been proposed involving custom security policies to control access privileges and roles but do not present a centralized and reusable architecture solution that can be customized for diverse SoC security challenges. Basak *et al.* [29] proposed an SoC-level solution (IIPS) for supply-chain security

[6]The Apple SE produces a modular security subsystem and is architecturally similar to CITADEL. Apple security enclave has a Root of Trust processor (similar to the compute enclave in the case of CITADEL), which controls a subsystem with other IPs. However, it is primarily a hardware-based key manager for data integrity and privacy [23].

Table 5: A Comparative Overview of CITADEL With Other Security Architectures

Architecture	Counterfeit Protection	RE# Protection	Lifecycle Management	Trojan Detection	Resource Isolation	Asset Provisioning
Apple Secure Enclave [22]	×	×	✓	✓	*	✓
OpenTitan [34]	✓	×	×	×	×	✓
ARM TrustZone [24]	×	×	×	✓	×	✓
E-IIPS [11]	✓	×	×	✓	×	×
SAP [31]	✓	×	×	×	×	✓
CHSM (3DIC) [32]	✓	✓	×	✓	×	✓
CITADEL	✓	✓	✓	✓	×	✓

Reverse Engineering.
* No public information available. Apple's security infrastructure is controlled by its software stack.

threats including counterfeiting, trojan attacks, etc. However, IIPS misses out on critical components like secure asset provisioning, device lifecycle management, device enrollment, handshaking, and secure boot control with the HOST processor. IIPS was further extended to E-IIPS [14] proposed an architecture to provide security against system-level trojans stemming from untrusted IPs affecting other IP cores or interconnect fabrics using run-time monitoring by security policies. It does not take into account other supply chain threats discussed above and also does not present a flexible architecture like CITADEL to augment its instance to support other security primitives. However, the solution proposed in E-IIPS can be integrated into CITADEL to provide security assurance against system-level trojans, as highlighted in Section 4. Moreover, both of these lack hardware and software-level SCM configurability. Another SoC security architecture and CAD framework [30] proposes a hardware patch and CAD flow for implementing security policies in-field on FPGAs, and lacks in providing any security against supply chain attacks like overproduction, counterfeiting, reverse engineering, illegal recycling, etc. While the idea of using security wrappers to tap low-level registers and scan for data is shared with CITADEL, their architectural integration, coordination, and access control are possible only using hooks provided by CITADEL. Note that each of these works primarily targets one specific threat class. Incorporating heterogeneous security primitives on an SoC level needs to account for the entire execution timeline of the SoC, and the various entanglements that come with it. None of the above solutions deal with such entanglements.

SAP [31] is a security architecture for enabling SoC authentication and secure asset provisioning in untrusted environments and chip authentication during deployment. However, SAP misses out on protection against reverse engineering attacks, which is a key component of supply-chain threats. Moreover, the architecture proposed in SAP is not extensible to provide support for reverse engineering solutions. Another key ingredient of security is its intervention during the boot process of the device, which is not mentioned in the paper. SAP was extended to CHSM [32], which does provide security assurance against reverse engineering attacks along with others, targeted for chiplets and 3DICs. However, re-purposing CHSM for SoC-based architectures would require significant and complex modifications in the architecture due to the difference in the attack surface between SoCs and chiplets. Further convoluting the transformation of the architecture for SoCs is that many of the security primitives used in CHSM exploit the 3D and chiplet properties, which are not present in SoC-based architectures. The paper also does not mention the integration of security provided by CHSM with the overall boot process of the underlying device.

PROTECTS [33], a secure protocol for provisioning security assets from an external entity (*i.e.* AMI) to the SoC in an untrusted fabrication and testing facility. This is only a small facet of the architecture proposed in CITADEL, which involves communication with the AMI at chip-birth for asset provisioning. PROTECTS, unlike CITADEL, however, does not solve the architectural problem of heterogeneous integration of diverse security primitives required to be deployed in an SoC environment to systematically protect against supply-chain attacks. To our knowledge, CITADEL is the first comprehensive architecture that comprehends requirements and constraints in integrating security solutions for an SoC, supports the entire SoC lifecycle, and accounts for control and coordination with the HOST processor, as well as protection of assets in its boundary.

7.2 Customized Security Solutions

There have also been customized solutions for individual supply-chain threats (*i.e.* overproduction, reverse engineering, etc). However, these approaches do not explore the question of integration requirements and system-wide coordination of these solutions in SoCs with custom security requirements.

There has been significant research on security protocols for safeguarding IP and IC designs against overproduction and counterfeiting attacks [35] [36] [37]. Such techniques include IC metering [38], Hardware Metering [39], CDIRs (Combating Die/IC Recovery) [40], and PUFs [41]. Techniques for mitigating reverse engineering attacks include camouflage, design obfuscation, split fabrication [42], etc. Camouflage is a layout-level mitigation strategy where multiple dummy contact points are added into the layout of the chip, altering the layout representation of the chip [43][44]. Design obfuscation techniques [7] include integrating additional logic in the design [45], and adding futile inputs and outputs to mask the target functionality. SCARe [46] is an SRAM-based technique to detect the aging of SRAM memory elements to determine the aging of SRAM cells in embedded devices. Other SRAM-based age detection techniques [47]. Other age detection techniques include EM on-chip imaging [48], and approximation of PUF signature variations due to aging [49].

8 Conclusion

In this paper, we introduced a security paradigm CITADEL, a skeletal security architecture for providing systematic security insertion and assurance in SoC designs for a wide variety of security use cases, and not limited to the supply chain. Architecturally, CITADEL is a plug-and-play subsystem composed of a microprocessor and a collection of custom IPs that can be integrated with an SoC design through a standardized set of interfaces. We outlined the methodology for deriving security architecture instances by following a threat-specific driven architecture philosophy and showcased the paradigm's viability for providing security against a wide variety of supply-chain threats. CITADEL has been designed from the ground up to provide resiliency against a spectrum of threats across the SoC lifecycle and enables systematic integration of a variety of security requirements without having to perturb the design functionality of the different SoC subsystems. Furthermore, CITADEL is built out of open-source components and exploits the RISC-V design ecosystem to achieve extensibility, configurability, and robustness. Finally, CITADEL achieves this with a very small hardware footprint, making it a viable security solution for low-power SoCs.

In future work, we will extend CITADEL with more use cases and threat models. We will also explore more optimization opportunities and the feasibility (and challenges) of integrating CITADEL with complex industrial SoCs.

References

[1] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. Aegis: Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th Annual International Conference on Supercomputing*, ICS '03, page 160–171, New York, NY, USA, 2003. Association for Computing Machinery.

[2] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 857–874, Austin, TX, August 2016. USENIX Association.

[3] Andrew Ferraiuolo, Andrew Baumann, Chris Hawblitzel, and Bryan Parno. Komodo: Using verification to disentangle secure-enclave hardware from software. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 287–305, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Rachel Selina Rajarathnam, Yibo Lin, Yier Jin, and David Z Pan. Regds: a reverse engineering framework from gdsii to gate-level netlist. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 154–163. IEEE, 2020.

[5] Policy Plans Office of Strategy. Combating trafficking in counterfeit and pirated goods. Jan 2020.

[6] Christopher Vega, Patanjali SLPSK, Shubhra Deb Paul, and Swarup Bhunia. Melpuf: Memory in logic puf, 2020.

[7] Md Moshiur Rahman and Swarup Bhunia. Practical implementation of robust state-space obfuscation for hardware ip protection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1–14, 2023.

[8] Yasaswy Kasarabada and Ranga Vemuri. Statelock: State transition based logic locking for sequential circuits. In *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, pages 171–176, 2020.

[9] Sandip Ray and Yier Jin. Security policy enforcement in modern soc designs. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 345–350, 2015.

[10] DARPA. Automatic implementation of secure silicon. 2020.

[11] Abhishek Basak, Swarup Bhunia, Thomas Tkacik, and Sandip Ray. Security assurance for system-on-chip designs with untrusted ips. *IEEE Transactions on Information Forensics and Security*, 12(7):1515–1528, 2017.

[12] Jinho Seol, Seongwook Jin, Daewoo Lee, Jaehyuk Huh, and Seungryoul Maeng. A trusted iaas environment with hardware security module. *IEEE Transactions on Services Computing*, 9(3):343–356, 2016.

[13] Daewon Kim, Yongsung Jeon, and Jeongnyeo Kim. A secure channel establishment method on a hardware security module. In *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 555–556, 2014.

[14] Abhishek Basak, Swarup Bhunia, and Sandip Ray. A flexible architecture for systematic implementation of soc security policies. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 536–543, 2015.

[15] Ezinam Bertrand Talaki, Olivier Savry, Mathieu Bouvier Des Noes, and David Hely. A memory hierarchy protected against side-channel attacks. *Cryptography*, 6(2), 2022.

[16] Yoav Weizman, Robert Giterman, Oron Chertkow, Maoz Wicentowski, Itamar Levi, Ilan Sever, Ishai Kehati, Osnat Keren, and Alexander Fish. Low-cost side-channel secure standard 6t-sram-based memory with a 1% area and less than 5% latency and power overheads. *IEEE Access*, 9:91764–91776, 2021.

[17] Jawad Haj-Yahya, Ming Ming Wong, Vikramkumar Pudi, Shivam Bhasin, and Anupam Chattopadhyay. Lightweight secure-boot architecture for risc-v system-on-chip. In *20th International Symposium on Quality Electronic Design (ISQED)*, pages 216–223, 2019.

[18] Hongil Ju, Yongsung Jeon, and Jeongnyeo Kim. A study on the hardware-based security solutions for smart devices. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 833–834, 2015.

[19] Atul Prasad Deb Nath, Kshitij Raj, Swarup Bhunia, and Sandip Ray. Soccom: Automated synthesis of system-on-chip architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(4):449–462, 2022.

[20] Brendon Chetwynd, Christopher Connelly, and Kyle Ingols. Common evaluation platform. In *MIT Lincoln Laboratory*, 2022.

[21] Paolo Matarazzo Vinay Pamnani. Trusted platform module technology overview.

[22] Apple. Secure enclave. 2021.

[23] Apple. Protecting keys with the secure enclave. 2021.

[24] ARM. Trustzone for cortex-a.

[25] Intel. White paper intel® software guard extensions (intel® sgx).

[26] Mohamed El-Hadedy and Xinfei Guo. Realse: Reconfigurable lightweight security engines for trusted edge devices. In *2022 IEEE 4th International Conference on Circuits and Systems (ICCS)*, pages 7–12, 2022.

[27] Manoj R Sastry, Ioannis T Schoinas, and Daniel M Cermak. Method for enforcing resource access control in computer systems, July 22 2014. US Patent 8,789,170.

[28] Markus Miettinen, Stephan Heuser, Wiebke Kronz, Ahmad-Reza Sadeghi, and N Asokan. Conxsense: automated context classification for context-aware access control. In *Proceedings of the 9th ACM symposium on Information, computer and Communications security*, pages 293–304, 2014.

[29] Xinmu Wang, Yu Zheng, Abhishek Basak, and Swarup Bhunia. Iips: Infrastructure ip for secure soc design. *IEEE Transactions on Computers*, 64(8):2226–2238, 2015.

[30] Atul Prasad Deb Nath, Sandip Ray, Abhishek Basak, and Swarup Bhunia. System-on-chip security architecture and cad framework for hardware patch. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 733–738, 2018.

[31] Md Sami Ul Islam Sami, Jingbo Zhou, Sujan Kumar Saha, Fahim Rahman, Farimah Farahmandi, and Mark Tehranipoor. Sap: Silicon authentication platform for system-on-chip supply chain vulnerabilities. In *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 109–119, 2024.

[32] Md Sami Ul Islam Sami, Tao Zhang, Amit Mazumder Shuvo, Md Saad Ul Haque, Paul E. Calzada, Kimia Zamiri Azar, Hadi Mardani Kamali, Fahim Rahman, Farimah Farahmandi, and Mark Tehranipoor. Advancing trustworthiness in system-in-package: A novel root-of-trust hardware security module for heterogeneous integration. *IEEE Access*, 12:48081–48107, 2024.

[33] Patanjali Slpsk, Jonathan Cruz, Sandip Ray, and Swarup Bhunia. Protects: Secure provisioning of system-on-chip assets in untrusted testing facility. In *2023 IEEE International Test Conference India (ITC India)*, pages 1–6, 2023.

[34] Google. Google opentitan security usecases.

[35] Sandip Ray and Yier Jin. Security policy enforcement in modern soc designs. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 345–350. IEEE, 2015.

[36] Sandip Ray. System-on-chip security assurance for iot devices: Cooperations and conflicts. In *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4. IEEE, 2017.

[37] Eric Peeters. Soc security architecture: Current practices and emerging needs. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[38] Sheng Wei, Ani Nahapetian, and Miodrag Potkonjak. Robust passive hardware metering. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 802–809. IEEE, 2011.

[39] Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Intellectual property metering. In Ira S. Moskowitz, editor, *Information Hiding*, pages 81–95, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[40] Xuehui Zhang, Nicholas Tuzzio, and Mohammad Tehranipoor. Identification of recovered ics using fingerprints from a light-weight on-chip sensor. In *DAC Design Automation Conference 2012*, pages 703–708, 2012.

[41] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 148–160, New York, NY, USA, 2002. Association for Computing Machinery.

[42] Yujie Wang, Pu Chen, Jiang Hu, and Jeyavijayan J V Rajendran. Routing perturbation for enhanced security in split manufacturing. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 605–510, 2017.

[43] Ronald P. Cocchi, James P. Baukus, Lap Wai Chow, and Bryan J. Wang. Circuit camouflage integration for hardware ip protection. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–5, 2014.

[44] Satwik Patnaik, Mohammed Ashraf, Ozgur Sinanoglu, and Johann Knechtel. Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4466–4481, 2020.

[45] Rajat Subhra Chakraborty and Swarup Bhunia. Harpoon: An obfuscation-based soc design methodology for hardware protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1493–1502, 2009.

[46] Zimu Guo, Xiaolin Xu, Md. Tauhidur Rahman, Mark M. Tehranipoor, and Domenic Forte. Scare: An sram-based countermeasure against ic recycling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(4):744–755, 2018.

[47] Ujjwal Guin, Wendong Wang, Charles Harper, and Adit D. Singh. Detecting recycled socs by exploiting aging induced biases in memory cells. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 72–80, 2019.

[48] Kai He, Xin Huang, and Sheldon X.-D. Tan. Em-based on-chip aging sensor for detection and prevention of counterfeit and recycled ics. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 146–151, 2015.

[49] Zimu Guo, Md. Tauhidur Rahman, Mark M. Tehranipoor, and Domenic Forte. A zero-cost approach to detect recycled soc chips using embedded sram. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 191–196, 2016.