Securing Wireless Integrated Circuits Against Supply Chain Attacks: SyncLock Demonstration

Alán Rodrigo Díaz Rizo, Hassan Aboushady, Haralampos-G. Stratigopoulos

Securing Wireless Integrated Circuits Against Supply Chain Attacks: *SyncLock* Demonstration

Alán Rodrigo Díaz Rizo, Hassan Aboushady, Haralampos-G. Stratigopoulos
Sorbonne Université, CNRS, LIP6, Paris, France

***Abstract*—Integrated Circuit (IC) supply chain attacks, such as piracy and counterfeiting, is a significant preoccupation for IC and System-on-Chip (SoC) designers. This article makes a practical demonstration of securing a wireless IC against such threats. The case study is an open-source IEEE 802.11 WiFi modem implemented on hardware using a Software Defined Radio (SDR) bladeRF board. The modem is secured with synchronization-based locking (*SyncLock*), a state-of-the-art locking scheme for RF transceivers. *SyncLock* disables the wireless communication between the modem and a WiFi-compliant receiver unless the correct secret key is loaded onto the modem.**

I. Introduction

The globalized and horizontally integrated supply chain of Integrated Circuits (ICs) is pivotal for technological progress, yet it is susceptible to security vulnerabilities. Such an IC supply chain, spanning design, fabrication, testing, packaging, and assembly, presents new risks as adversaries can exploit vulnerabilities throughout the IC life cycle. Notably, the reliance of modern System-on-Chip (SoC) designs on third-party IP cores and outsourced fabrication raises significant concerns on the protection of intellectual property rights of ICs. IP/IC piracy and complete SoC counterfeiting are major preoccupations nowadays for the design houses. Researchers have explored defenses, ranging from techniques targeting specific single threats, e.g., an untrusted foundry or reverse-engineering, to holistic end-to-end solutions, such as locking, to fight against the complete landscape of IC supply chain attacks.

Locking, performed by the legitimate IP, IC, or SoC designer, embeds into the original design a circuit, namely lock mechanism, controlled by a key, which is typically a binary string. For the correct key, the lock mechanism is transparent and the intent functionality is established, while for an incorrect key the functionality is corrupted. The lock mechanism is mingled with the original design and the key is never shared with untrusted parties. The IC is securely activated after fabrication by a secure Key Management Scheme (KMS). The most common KMS consists in storing the key in a Tamper-Proof Memory (TPM), whose content is erased when detecting a probing attempt.

Locking was originally proposed for digital ICs [1], a.k.a. Logic Locking (LL) or logic encryption. The basic idea is to insert into the design key-gates controlled by the key-bits, such that when the correct key is employed the key-gates become transparent. Then, its application was extended to Analog Mixed-Signal (AMS) ICs by leveraging LL to lock individual blocks in the digital section of an AMS IC [2], [3]. Locking techniques specific to AMS ICs include biasing [4], [5], calibration [2], [6]–[8], and layout-level locking [9]. Biasing locking aims at regulating the bias generation with the key, e.g., by obfuscating the geometry of bias transistors. However, effective counterattacks have been proposed [10]–[12]. Calibration locking controls the tuning of the circuit with the key. However, it requires many tunable parameters to resist brute-force attacks, and mandates that the calibration algorithm is complex to be worked out by the attacker. Layout-level locking replaces the transistor layout with a parallel connection of different layout structures exhibiting different Layout-Dependent Effects (LDEs), with the correct key switching on the combination of structures for which the circuit is optimized to work. There exist also key-less layout obfuscation techniques that protect only against an untrusted foundry [13] or only against reverse engineering of a legally purchased chip [14], [15].

For wireless ICs comprising an RF transceiver, two approaches have been proposed, namely LL of the digital baseband Physical (PHY) layer [16] and synchronization-based locking (*SyncLock*) [17]. *SyncLock* is a locking technique specific to RF transceivers. Unless the correct key is used, the receiver is not synchronized with the transmitter, thus the communication link is not established. The advantage of *SyncLock* compared to standard LL is that it is resilient to the various counterattacks to LL. These counterattacks aim at either extracting the correct key with few queries or identifying and removing the lock mechanism with a netlist analysis. To perform the query, it is required to apply different inputs to an oracle chip (i.e., with the correct key stored), while the netlist analysis is typically based on tracing the lock mechanism by its connection to the TPM. *SyncLock* thwarts these attacks because the input to the locked section is fixed and because the lock mechanism has two parts, one connected to the TPM and one that is spatially separated from the TPM.

In this work, we present a first practical hardware demonstration of *SyncLock*. We first implemented *SyncLock* into an open-source IEEE 802.11 WiFi modem. We show that *SyncLock* prevents the locked modem to synchronize and exchange data with a WiFi standard-compliant smartphone. The effect of *SyncLock* on the communication link is monitored by a third device that performs spectrum analysis. To promote reproducible research, the experiment files are made publicly accessible.

The remainder of this article is organized as follows. In Section II, we provide an overview of the principle of op-

eration of *SyncLock* and its security against counter-attacks. In Section III, we describe the hardware platform used for the demonstration. In Section IV, we describe the *SyncLock* implementation and the resultant overhead. In Section V, we present the demonstration. Section VI concludes this article.

II. *SyncLock* PRINCIPLE OF OPERATION

We showcase a demonstration of *SyncLock* for a wireless IC implementing the WiFi standard. However, it should be pointed out that *SyncLock* is applicable to any communication protocol whose synchronization process is based on correlation, such as Bluetooth or Zigbee. The WiFi standard employs the Physical Protocol Data Units (PPDU) frame format [18]. A PPDU generally consists of several Orthogonal Frequency-Division Multiplexing (OFDM) symbols, each consisting of subcarriers modulated as BPSK, QPSK, 16-QAM, 64-QAM, or 256-QAM. These symbols are divided into preamble and data. The preamble field comprises two different training symbol sequences: a Short Training Sequence (STS) and a Long Training Sequence (LTS). The data field comprises a header symbol, with information of the length of the PPDU and the modulation and coding scheme of the following OFDM symbols, and the data symbols, a.k.a. payload data.

SyncLock is based on the corrupt-and-correct principle with its lock mechanism being composed of two spatially separated operations. The corruption, driven by a corruption key, denoted by key_{h-c}, hardcoded into the design, impacts the synchronization sequence, i.e., the STS, sent by the transmitter and expected by the receiver, thus disrupting the communication. The correction, driven by the secret key, denoted by key, sourced from the KMS, compensates for the corruption and fixes the STS, thus restoring communication.

Fig. 1 illustrates a simplified block diagram of a WiFi RF transceiver with *SyncLock*. Fig. 1 shows only the necessary blocks to explain the *SyncLock* principle of operation. Connections with dashed arrows symbolize that there are other PHY layer blocks between them, such as the creation and addition of cyclic prefixes. The corruption operation is located inside the PPDU generation block and disturbs the STS that will be prepended to every generated transmission frame. The correction operation is located inside the STS generation block and alters the nominal STS based on the key such that downstream corruption is compensated.

As the corruption module is isolated from the key inputs, an attacker possessing the netlist cannot locate it by analyzing the netlist. This is because the netlist is non-annotated and the small corruption module is disguised within a sea of gates. Moreover, the STS generation block has a fixed input, i.e., the nominal STS, thus query attacks on a oracle chip to identify the key are not applicable.

III. HARDWARE PLATFORM FOR DEMONSTRATION

The hardware demonstrator employs the Software Defined Radio (SDR) bladeRF board from Nuand [19]. It is composed of a Cypress FX3 microcontroller, a fully programmable Cyclone V Field Programmable Gate Array (FPGA) from

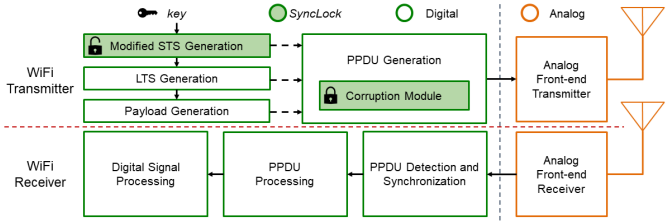

Fig. 1: Simplified block diagram of a WiFi RF transceiver with *SyncLock*.

Intel, and an RF transceiver AD9361 from Analog Devices. We chose the open-source *bladeRF-wiphy* project from Nuand as our case study [20]. *bladeRF-wiphy* is an IEEE 802.11 compatible SDR VHDL modem that connects the baseband PHY layer and the Medium Access Control (MAC) layer. Working together, they enable the SDR to exchange data with any standard-compliant device, effectively functioning as a WiFi access point.

The MAC layer operates in software on the host PC using the Linux 802.11 frame manager `mac80211`. The VHDL modem runs in the FPGA embedded in the bladeRF board and implements the baseband PHY layer of the WiFi protocol, i.e., it enables the bladeRF board to modulate and demodulate WiFi PPDUs. The PPDUs are then transmitted and received through the Analog Front-End (AFE) of the board, as shown in Fig. 1. More specifically, the transmitter's part of the PHY layer prepares the PPDUs and transmits them using the AFE transmitter of the board. At the receiver part, the RF signal is recovered using the AFE receiver of the board. The PHY layer constantly acquires signals and awaits the beginning of a PPDU, i.e., the STS. The receiver uses the STS to estimate the start of the PPDU, i.e., to synchronize the transmitter with the receiver. Finally, the synchronized PPDUs are demodulated and then processed at the host PC by the MAC layer.

IV. *SyncLock* IMPLEMENTATION

The *SyncLock* mechanism resides only in the transmitter part, namely in the *wlan_tx* module of *bladeRF-wiphy*. First we perform a Register-Transfer Level (RTL) analysis of *wlan_tx* to identify where the STS is generated, how it is generated, and finally how it is prepended to the rest of the PPDU. The module that generates the STS is called *wlan_tx_short*. It creates the time-domain sequence defined by the WiFi standard based on stored and fixed values. The *wlan_sample_buffer* module forms the PPDU with the STS, the LTS, and the baseband IQ samples of the payload, and sends it to the Digital-to-Analog Coverter (DAC) of the AFE.

A destination in the datapath of the *wlan_tx_short* module is modified with an XOR cipher to control the nominal STS, denoted by STS_{nom}, with the secret key. The output of the modified *wlan_tx_short* module is given by

$$STS_{faulty} = STS_{nom} \oplus key. \quad (1)$$

The corruption module is embedded into the datapath of the *wlan_sample_buffer* module. It utilizes the hardcoded key and a non-linear function $f(\cdot)$ which, for example, can be a

TABLE I: Overhead when adding *SyncLock* into *bladeRF-wiphy*.

Logic utilization	[20]	This work
Adaptive Logic Modules (ALMs)	31%	**32%**
Total registers	55634	**55828**
Total pins	77%	77%
Total block memory bits	17%	17%
Total RAM Blocks	27%	27%
Total DSP Blocks	29%	29%
Total PLLs	50%	50%

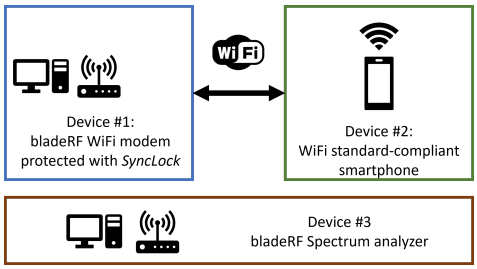

Fig. 2: Demonstration setup.

circular shift operation (i.e., bitwise rotation), a bitwise logical operation, a bit scrambling, a bit or byte substitution, etc. The input STS to the *wlan_sample_buffer* module originating from the *wlan_tx_short* module, i.e., STS_{faulty}, is modified to result in the following output STS for the *wlan_sample_buffer* module

$$STS_{out} = STS_{faulty} \oplus f(STS_{out}, key_{h-c}). \quad (2)$$

Combining Eqs. (1)-(2) and using the associative property $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ of the XOR function, the system equation becomes

$$STS_{out} = STS_{nom} \oplus (key \oplus f(STS_{out}, key_{h-c})). \quad (3)$$

While STS_{nom} is known and STS_{out} can be measured, the attacker cannot extract the secret key since $f(\cdot)$ and key_{h-c} are unknown and hardcoded. The interested reader is referred to [17] for a more detailed description of *SyncLock*.

The resulted *SyncLock*-enabled *bladeRF-wiphy* is then implemented into the FPGA of the bladeRF board. STS_{nom} is a 512-bit word, thus the key can have a size of up to 512 bits, easily thwarting a brute-force attack. The resources overhead of the implementation compared to the benchmark *bladeRF-wiphy* is shown in Table I. In terms of logic utilization, only 194 registers and 1% of the total Adaptive Logic Modules (ALMs) were added, thus the area and power overhead considering an ASIC implementation are practically negligible.

V. Demonstration

The demonstration employs three devices as depicted in Fig. 2. Device #1 is a bladeRF board that serves as a WiFi modem with the *SyncLock* mechanism embedded into its PHY layer according to the methodology presented in Section IV. Device #2 is a WiFi standard-compliant smartphone. The experiment consists of Device #2 trying to establish a connection with Device #1 via WiFi with their distance being one meter. When the connection is established, Device #2 interacts with Device #1 downloading images and text from the host computer connected to Device #1. Device #3 is a second bladeRF board that serves as a spectrum analyzer. It runs a GNU Radio Companion script to monitor the central frequency band where the WiFi communication is happening and the two Devices #1 and #2 exchange data. The spectrum analyzer is tuned to a central frequency of 5.7 GHz with a bandwidth of 20 MHz, having a receiver gain of 60 dB and a sampling rate at 40 Mega-samples per second (Msps).

We demonstrate two scenarios, namely Device #1 is unlocked having the secret key loaded and Device #1 is locked having a random incorrect key loaded. Fig. 3 shows three different monitoring states by Device #3.

State *S0* in Fig. 3(a) corresponds to Device #1 being turned off. In this case, the magnitude level of the received signal by Device #3 is below -66 dB and no RF activity is detected.

State *S1* in Fig. 3(b) corresponds to Device #1 being turned on and having the correct key loaded. Device #2 detects Device #1 using the WiFi protocol and is synchronized to Device #1 starting downloading data transmitted from Device #1. As seen from Fig. 3(b), Device #2 captures the communication. The magnitude level of the received signal is above -40 dB and RF activity is detected. By uploading the correct key into Device #1, the *SyncLock* mechanism becomes transparent to the normal operation of Device #1. The Bit Error Rate (BER) is identical to the case where Device #1 has no locking mechanism embedded [17].

State *S2* in Fig. 3(c) corresponds to Device #1 being turned on but having an incorrect key loaded. In this case, Device #1 is sending an acknowledgement to make a handshake with other wireless devices. Device #3 is capable of detecting this RF activity. As can be seen from Fig. 3(c), the magnitude level of the received signal is above -60 dB. However, as Device #2 was not able to detect Device #1 using the WiFi protocol due to locking, Device #3 does not change to state *S1*. More specifically, due to the corrupted preamble field STS in the PPDUs sent by Device #1, Device #2 cannot detect the start of the PPDU and cannot synchronize with Device #1. If an incoming sample were chosen at random, it would most likely not be the first sample of the STS. If a random sample was forced as the start of a PPDU, the demodulator in Device #2 would process the contents of the PPDU with meaningless values and, as a result, Device #2 would be unable to establish the connection. In this case, the BER is proven to be maximal [17].

The interested reader can download the demonstration files from this link: *https://nuage.lip6.fr/s/5dF7CKrtBa73ZCS*. The folder includes: (a) The FPGA bitstreams containing the VHDL modem implementation with the secret key and an incorrect key; (b) The GNU Radio Companion script for the spectrum analyzer; and (c) a video of the demonstration.

VI. Conclusion

SyncLock is a state-of-the-art locking scheme for wireless ICs protecting against piracy, counterfeiting, reverse-engineering, and unauthorized use. In this article, we described

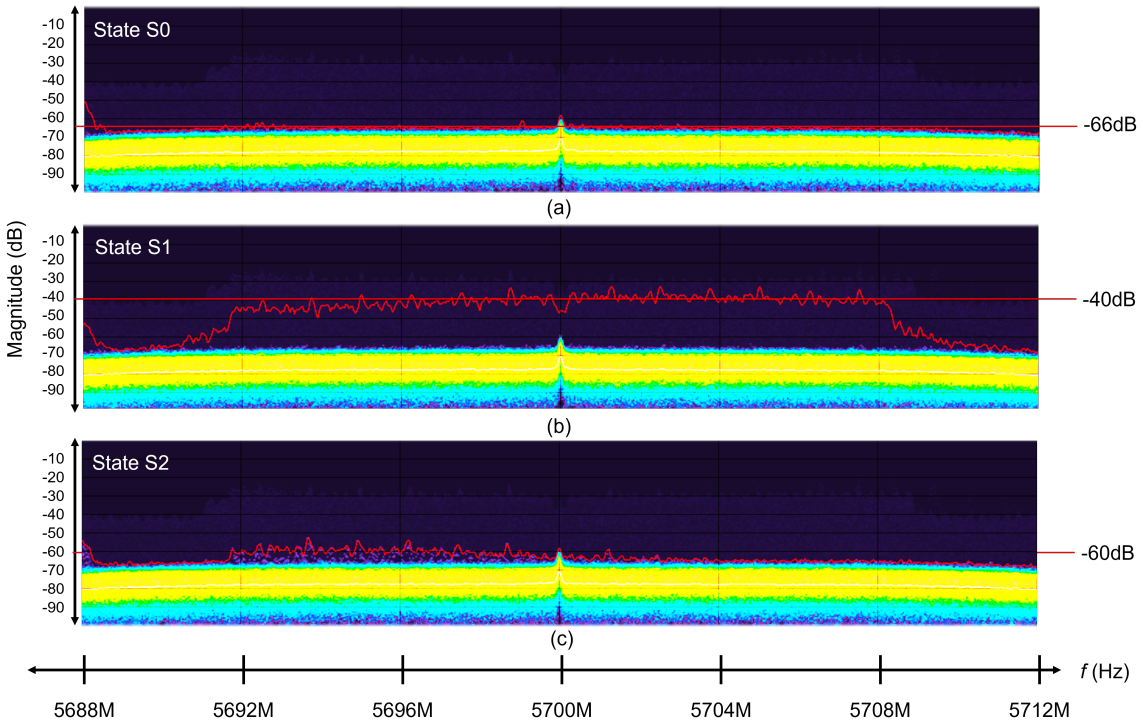

Fig. 3: Spectrum analyzer (Device #3) output.

a first practical hardware demonstration of *SyncLock*. An open-source IEEE 802.11 WiFi VHDL modem is modified to incorporate *SyncLock* and is set to establish communication with a WiFi standard-compliant smartphone. The communication link is continuously monitored via a spectrum analyzer. We showed that unless the correct key is loaded into the modem, the communication link crashes.

REFERENCES

[1] K. Z. Azar, H. M. Kamali, F. Farahmandi, and M. Tehranipoor, *Understanding Logic Locking*, Springer, 2023.

[2] N. G. Jayasankaran, A. Sanabria Borbon, E. Sanchez Sinencio, J. Hu, and J. Rajendran, "Towards provably-secure analog and mixed-signal locking against overproduction," *IEEE Trans. Emerg. Top. Comput.*, vol. 10, no. 1, pp. 386–403, Jan.-Mar. 2022.

[3] J. Leonhard *et al.*, "Digitally-assisted mixed-signal circuit security," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2449–2462, Aug. 2022.

[4] J. Wang, C. Shi, A. Sanabria-Borbon, E. Sánchez-Sinencio, and J. Hu, "Thwarting analog IC piracy via combinational locking," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2017.

[5] V. Rao and I. Savidis, "Performance and security analysis of parameter-obfuscated analog circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 12, pp. 2013–2026, Dec. 2021.

[6] S. G. Rao Nimmalapudi, G. Volanis, Y. Lu, A. Antonopoulos, A. Marshall, and Y. Makris, "Range-controlled floating-gate transistors: A unified solution for unlocking and calibrating analog ICs," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar 2020.

[7] M. Elshamy, A. Sayed, M.-M. Louërat, H. Aboushady, and H.-G. Stratigopoulos, "Locking by untuning: A lock-less approach for analog and mixed-signal IC security," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 12, pp. 2130–2142, Dec. 2021.

[8] M. Tlili, A. Sayed, D. Mahmoud, M.-M. Louërat, H. Aboushady, and H.-G. Stratigopoulos, "Anti-piracy of analog and mixed-signal circuits in FD-SOI," in *Proc. 27th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Jan. 2022, pp. 423–428.

[9] M. J. Aljafar, F. Azais, M.-L. Flottes, and S. Pagliarini, "Utilizing layout effects for analog logic locking," 2024, [online] Available: http://arxiv.org/abs/2401.06508.

[10] N. G. Jayasankaran, A. Sanabria-Borbón, A. Abuellil, E. Sánchez-Sinencio, J. Hu, and J. Rajendran, "Breaking analog locking techniques," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 10, pp. 2157–2170, Oct. 2020.

[11] R. Y. Acharya, S. Chowdhury, F. Ganji, and D. Forte, "Attack of the genes: Finding keys and parameters of locked analog ICs using genetic algorithm," in *Proc. IEEE Int. Symp.Hardw. Oriented Secur. Trust (HOST)*, Dec. 2020, pp. 284–294.

[12] J. Leonhard, M. Elshamy, M.-M. Louërat, and H.-G. Stratigopoulos, "Breaking analog biasing locking techniques via re-synthesis," in *Proc. 26th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Jan. 2021, p. 555–560.

[13] M. R. Muttaki, H. M. Kamali, M. Tehranipoor, and F. Farahmandi, "PALLET: Protecting analog devices using a last-level edit technique," in *Proc. IEEE Phys. Assurance Inspection Electron. (PAINE)*, Oct. 2023.

[14] A. Ash-Saki and S. Ghosh, "How multi-threshold designs can protect analog IPs," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 464–471.

[15] J. Leonhard, A. Sayed, M.-M. Louërat, H. Aboushady, and H.-G. Stratigopoulos, "Analog and mixed-signal IC security via sizing camouflaging," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 5, pp. 822–835, Jul. 2021.

[16] A. R. Díaz-Rizo, J. Leonhard, H. Aboushady, and H. Stratigopoulos, "RF transceiver security against piracy attacks," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 69, no. 7, pp. 3169–3173, Jul. 2022.

[17] A. R. Díaz-Rizo, H. Aboushady, and H.-G. Stratigopoulos, "Anti-piracy design of RF transceivers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 1, pp. 492–505, Jan. 2023.

[18] IEEE, "IEEE standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, 2016.

[19] Nuand, "SDR bladeRF 2.0 micro xA9," [online] https://bit.ly/3z2QV1N.

[20] Nuand, "Open-source IEEE 802.11 compatible software defined radio VHDL modem (bladeRF-wiphy)," [online] Available: https://github.com/Nuand/bladeRF-wiphy/.