Journal of
Sensor and Actuator Networks

MDPI

Review

Hardware Security in IoT Devices with Emphasis on Hardware Trojans

Simranjeet Sidhu [1], Bassam J. Mohd [2] and Thaier Hayajneh [1,*]

[1] Fordham Center for Cybersecurity, Fordham University, New York, NY 10023, USA
[2] Department of Computer Engineering, Hashemite University, Zarqa 13115, Jordan
* Correspondence: thayajneh@fordham.edu; Tel.: +1-212-636-7785

Received: 30 June 2019; Accepted: 1 August 2019; Published: 10 August 2019

Abstract: Security of IoT devices is getting a lot of attention from researchers as they are becoming prevalent everywhere. However, implementation of hardware security in these devices has been overlooked, and many researches have mainly focused on software, network, and cloud security. A deeper understanding of hardware Trojans (HTs) and protection against them is of utmost importance right now as they are the prime threat to the hardware. This paper emphasizes the need for a secure hardware-level foundation for security of these devices, as depending on software security alone is not adequate enough. These devices must be protected against sophisticated attacks, especially if the groundwork for the attacks is already laid in devices during design or manufacturing process, such as with HTs. This paper will discuss the stealthy nature of these HT, highlight HT taxonomy and insertion methods, and provide countermeasures.

Keywords: hardware security; security of IoT devices; hardware Trojans; Trojan detection techniques; side-channel analysis; hardware Trojan-based side-channel attacks; hardware Trojan countermeasures

1. Introduction

The extremely complex IoT ecosystem involves a large number of interconnected IoT devices, and this number is expected to significantly increase within a few years. The security of the IoT environment must be given a high priority while developing, configuring, and updating the devices to ensure optimal performance throughout their lifetime. Protection of IoT devices against cyber-attacks can help achieve a high level of confidentiality, integrity, and availability with authorized access to the users. Security of IoT devices is a work in progress, and proliferation of these devices at an alarming rate is making it more challenging to secure them. IoTs are an attractive target for adversaries because of their easy accessibility, vulnerabilities, and the quality of data they hold. IoT devices are usually connected to other devices on the same network, which puts all other devices at risk if one of them gets compromised. Therefore, security of these devices regardless of their size is paramount.

Currently, these devices are very vulnerable to various attacks, which can lead to huge damage potential. The distributed nature of IoT devices with their limited computational and storage capabilities make them easily exploitable. Devices that are easily accessible, such as sensors and actuators are prone to physical attacks, such as being tampered with in order to provide incorrect data to the nodes or being sent commands from unauthorized sources. For example, a tampered security camera may always show a still image, allowing intruders in without being detected. The healthcare IoT platform has also been growing over the last few years. Patients wear wireless sensing IoT devices which collect private health-related information. Security of these devices and the data collected by them is extremely important and very challenging [1]. Recent cases of ransomware attacks on implanted medical devices, such as the WannaCry ransomware attack in May 2017, targeted devices running Windows OS and encrypted files on the hard drive making it inaccessible for users. The adversaries

demanded a ransom payment from victims in return of decrypting the files [2]. Such cases are a prime example that digital security has become a matter of life and death now.

Using software security alone to protect these devices is not adequate, as the threats in these distributed systems are not confined to the software/network layer only. There is a big demand for implementation of security at the hardware level. Without a secure hardware, the IoT devices cannot have a firm foundation to build a secure infrastructure on. IoT devices have a long deployment and usage lifetime without a lot of human interference or management, unlike Personal Computers (PCs) that are accessed and updated on a regular basis. The expectation from IoT devices such as security cameras is to be deployed for years without any interruption. Hardware security is essential to ensure secure deployment of these devices throughout their lifetime, especially for the devices that are not easily accessible to be updated regularly. IoT devices have many applications in our lives these days. At least one IoT device can be found in the majority of networks nowadays, from our personal home network to national defense and the medical sector. Therefore, security of these devices is paramount for individual as well as national security.

With the advancement in technology, the sophistication of attacks is also increasing. Malicious insertions and modifications can be made during chip design. A small microchip called Integrated Circuit (IC) is an integral part of any smart device that can perform basic data processing and communication. These malicious modifications are done in such a professional manner that they are hard to detect during the testing phase. The objectives of this paper are to:

A. Emphasize the importance of implementing hardware security among IoT devices.
B. Discuss challenges in securing hardware for IoT devices.
C. Highlight threats to hardware and introduce Hardware Trojan (HT).
D. Provide detailed taxonomy of Hardware Trojan and discuss some insertion methods.
E. Provide countermeasures, with focus on HT detection techniques and Design for Trust.

The paper is organized in the following way: Section 2 discusses challenges being faced in implementing hardware security in IoT devices, Section 3 sheds some light on hardware security threats, which covers HT and side-channel attacks. Section 4 presents countermeasures that can be used for strengthening hardware security, and finally Section 5 provides a conclusion for the paper.

2. Challenges

The main objectives of security, highlighted by various models are known to be confidentiality, integrity, authentication, authorization, nonrepudiation, and availability. Implementing different cryptographic mechanisms can help us achieve these security characteristics. However, it is very challenging to implement these cryptographic algorithms in IoT devices. A lot of IoT devices are extremely small in size and fall into the category of low-resource devices (LRD) or constrained-resource devices (CRD). Their physical limitation allows them to possess low-power resources only, which makes adding more layers of security difficult. Constrained resources, such as limited central processing unit (CPU) and memory, limit their ability to process information and complex security algorithms. There is a serious tradeoff between security and resources when it comes to IoTs. Confidentiality is most significantly sacrificed due to constraints in size and power of IoTs [3]. Implementation of Public Key Infrastructure (PKI) can provide confidentiality and integrity. However, the encryption process with public key demands computational and memory resources that are beyond the capabilities of many IoT devices. Also, PKI requires the devices to be continuously updated as the certificates expire. Updating IoT devices is another challenge, especially in really small devices which do not have any user interface. Moreover, some IoT devices are deployed remotely in inaccessible areas and cannot be updated without human intervention [4]. Implementing encryption in these low-resource devices is a current field of interest for researchers. References [5–8] present various methods to improve lightweight block cipher for low-resource IoT devices for securing these devices, data, and communications between them.

Attestation along with Hardware Security Managers (HSM) is used to achieve integrity. HSM provides security of authentication keys [4]. These keys are usually encrypted with a human's password or another identification parameter, and thus require human intervention. This is difficult to implement in cases of inaccessible and unattended IoT devices. Thus, another method needs to be implemented to protect authentication keys. An alternative way to protect the keys could be storing them in a dedicated hardware storage. However, firmware modification of the device could lead to authentication keys being read by modified firmware, and they might fall into hands of an adversary [4].

Authentication and identification of users in IoTs is also significantly important. If an attacker compromises authentication, they can get access to the system as a legitimate user and can launch various further attacks without even being detected. However, authentication has been another major challenge in IoTs. Current authentication methods include username/password, digital certificates, shared keys, or biometric credentials. It is anticipated that IoTs as pervasive will remove many physical interaction interface mediums through which username/passwords are passed [3]. Furthermore, verifying identity can be challenging in the mobile environment of multiple IoT devices. Different users move their IoT devices through different architectures and infrastructures provided by different service providers. This issue of anonymity makes it difficult to achieve accountability.

To attain effective access control is another challenge in the IoT domain. It is difficult to use well-known access-control models such as role-based access control (RBAC) and attribute-based access control (ABAC) for low-powered devices [9]. Furthermore, access control requires the concept of authentication and identity, which is already a challenge in IoTs, as discussed earlier. The resource-constrained nature of IoTs could lead to energy consumption and resource exhaustion attacks, which can lead to DoS attacks, affecting availability as well. Nonrepudiation is another challenge in IoTs, which cannot be achieved without proper attestation. Proper attestation is already hard to achieve in IoTs.

There are other challenges that add to already existing issues. Hardware threats are increasing at a higher rate than the implementation of hardware security controls. Some IoT devices are deployed in easily accessible, unfriendly, and less supervised areas, which allows attackers to physically get a hold of the object and tamper it. Hardware vulnerabilities are very difficult to detect. An example of it is the attack suffered by Ivy Bridge intel processors, that included transistors of chips being doped to change the random number generator (RNG). RNG is an important base for encryption systems, and the attack resulted in a fixed output of RNG [10]. Detecting such a small modification in a circuit is very challenging.

The globalization of chip manufacturing is another challenge that adds more problems for security at the hardware level. The IC supply chain is spreading around the globe to meet the needs of advancing design complexity of system-on-chip (SoC), high cost of fabrication, and to reduce the time-to-market. More companies are relying on third-party fabrication services and third-party IP cores [11]. Being dependent on other vendors has given rise to more security concerns. Attackers can get access to the IC supply chain and insert malicious logic or design flaws during any design stage. Such threats can force the IC to malfunction during its runtime. These companies are located across the world, making it difficult to control the design process. Hardware Trojan is one such threat and the main reason for the emergence of hardware security. Hardware Trojans are stealthy and difficult to detect as they have the capability to remain inactive for a long time, even during the testing phase, and may activate at a later time while running when they are triggered [11,12].

3. Hardware Security Threats

It has been realized that true security of IoT devices can only be achieved by securing the underlying hardware of these devices. Hardware security begins with IC that is embedded in these devices. Backdoors have been reported in ICs used in weapons control systems, nuclear power plants, and public transportation systems [11]. Security is usually an afterthought in IC design, and preference is given to cost, performance, and reliability [11].

As discussed in the previous section, the globalization of IC design, manufacturing, and distribution in the supply chain is an emerging problem. The need to meet the increased demand for IC boards has pushed companies to outsource the production of different components to various cheaper factories around the world, which introduces various threats. The process of designing an IC usually combines designing some components inhouse and some Intellectual Property (IP) designs being procured from third-part design houses. Then, a blueprint of these designs is sent to a foundry for manufacture and testing [11]. Sometimes ICs are also tested at third-party facilities, after which they are packaged and sold. Throughout this process, many things may go wrong. Hardware Trojans (HT) or backdoors could be inserted by adversaries during any design stage. Illegal piracy of IP can be done. Excess ICs may be built by a malicious foundry and sold in the gray market. Reverse engineering of the IC/IP design is possible. Side-channel attacks can be made to extract the secret information. Original design or components can be cloned illegally by an adversary [11].

Reference [13] describes "Pyramid of Pain" as shown below in Figure 1, which is based on Cisco's IoT reference model, and it is evaluated from a vulnerability perspective of the IoT system. The most vulnerable part of the IoT system with the least impact if suffered an attack is placed on the top of the pyramid. Sensors sit on the top as they are the most vulnerable, being the most accessible part of the IoT ecosystem. The next vulnerable part is the communication between the sensors and accumulated data by the sensors. The attackers can get access to this data through sensors or by getting into the network. Then comes the hardware abstraction and firmware, which stipulate Application Programming Interface (API) for the interaction between the application and the data. Lastly, the least accessible part, at the bottom of the pyramid is the hardware platform such as SoC, FPGAs (field-programmable gate arrays), DSP (digital signal processors), etc. Even though the hardware platform is at the bottom of the pyramid, it is the one that deserves major attention because it has the highest damage-impact to a system in case of an attack. It can be said that hardware is the foundation of the IoT system and the most pain-causing part in case of a cyber-attack incident. Therefore, IoT security should begin with hardware security. The greatest threat to hardware is HT, which is why it is the main focus of this paper.

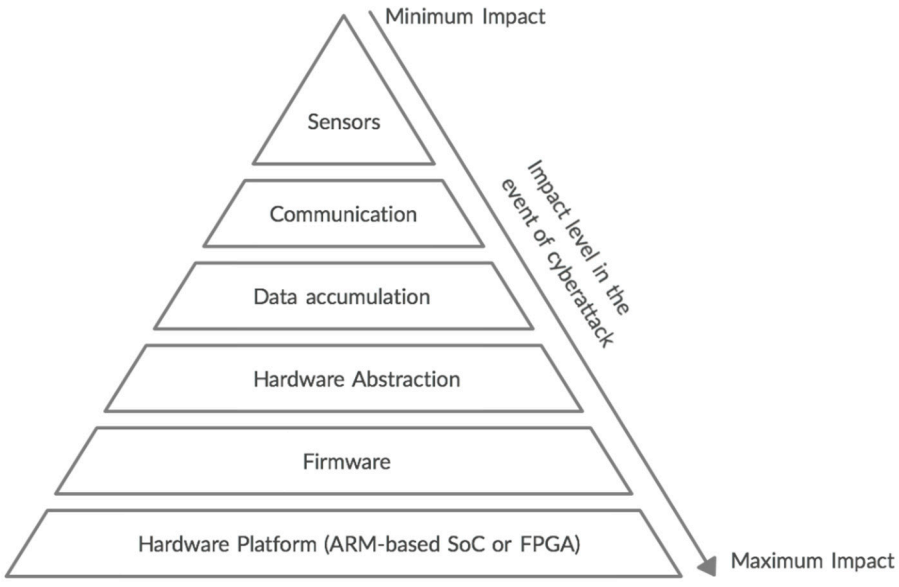

Figure 1. "Pyramid of Pain", based on vulnerability analysis.

3.1. Hardware Trojan

An HT is a deliberate malicious insertion or alteration to the existing circuit, resulting in change of functionality of the circuit when activated. Adversaries insert HT in an integrated circuit to make it less reliable or to completely destroy it. The threats posed by HT are severe throughout the device's

lifetime as it is embedded within the device's lowest level. HT can infect any kind of IC, such as application-specific integrated circuits (ASIC), system-on-chip (SoC), field-programmable gate arrays (FPGAs), and digital signal processors (DSP) [13]. HTs impede security measures existing in the chip, resulting in the infected chip to operate erroneously. The HT can lead to information leakage, bypass of the security of the system, or even complete destruction of the chip.

HTs are a direct threat to already vulnerable IoT. The behavior of an HT cannot be changed once it is inserted. Unlike software Trojans, HTs cannot be eliminated simply by a firmware update, which is why they are very detrimental and more challenging to remove. HTs may cause many damaging effects to the IoT ecosystem, such as information leakage, denial of service (DoS) attacks, service degradation, and failure of the device. Some of the attractive targets for adversaries are IoT devices used in military, aerospace, financial, telecommunication, and commercial industries [14]. National security may depend on the security of these IoT devices.

Another daunting characteristic of HTs is that they are very difficult to detect, especially in their dormant state. Reference [15] demonstrated insertion of a malicious circuit on an RC4 stream cipher for easy cryptanalyzing of the generated stream by lowering random encryption. The analysis showed that only a few logical gates are enough to trigger an HT without any noticeable variations in the circuit parameters. These small variations in circuit parameters after Trojan insertion make the HT very hard to detect during testing phase.

3.2. Hardware Trojan Taxonomy

To grasp the understanding of HTs, it is important to discuss its taxonomy. Proper classification of hardware trojans forms a good basis for the implementation of appropriate solution techniques. Reference [16] classifies HTs based on the activation property of Trojans: Always On or Triggered (internally triggered or externally triggered). Reference [17] does the classification based on the Physical Activation and Action characteristics of Trojans, where action represents the actions taken by the Trojan once it i activated. Reference [18] partitions HTs into two types based on trigger and payload mechanisms. Payload is classified further into digital, analog, and other. Reference [19] classifies HT based on insertion phase, abstraction level, activation, effects, and location. We have developed a combination and expanded version of HT taxonomy, which is shown in Figure 2.

Physical: First, classification of HT taxonomy is based on physical attributes of the trojan, which is further distributed into size, type, distribution, and structure.

- *Type*: Type distributes Trojan into functional and parametric classes. Trojans that are introduced through addition or removal of gates or transistors fall into the functional category, whereas trojans introduced by modifying existing wires or logic belong to the parametric category [17].
- *Size*: Size of a trojan depends on the number of components in the chip added or deleted. The smaller the trojan, the higher its probability for activation [17].
- *Distribution*: This signifies the location of the Trojan in the chip. If a Trojan's components are topologically close in physical layout of the chip, it is categorized into tight distribution. If a Trojan is dispersed across the layout of the chip, it falls into the loose distribution class [17].
- *Structure*: Adversaries try to make sure that insertion of a Trojan should not change the physical layout of the circuit in order to evade detection [17].

Insertion Phase: This signifies the stage at which the Trojan may be inserted during design and manufacture of an IC. The various stages that offer an entry point for HT insertion are specification, design, fabrication, testing, and assembly [13].

Activation: HTs can be activated by the occurrence of certain events inside or outside of the system. However, some Trojans do not require a trigger to be activated and can disrupt the chip's function any time. Such Trojans fall into Always On category. Others require an internal or external trigger in order to be activated. The main goal of the trigger HT is not to be continuously active, as it may make it capable of being detected [15]. The internal trigger could sense environment/conditions around

the device, being caused by a physical sensor on the IoT device that detects things such as humidity, temperature, etc. An internal logical state or counter value may also activate a Trojan. Some Trojans are designed to be activated at a certain time or after a specific amount of delay. The external triggers include instructions sent remotely after taking advantage of weak network security or an input entered by the user without the user's knowledge [16].

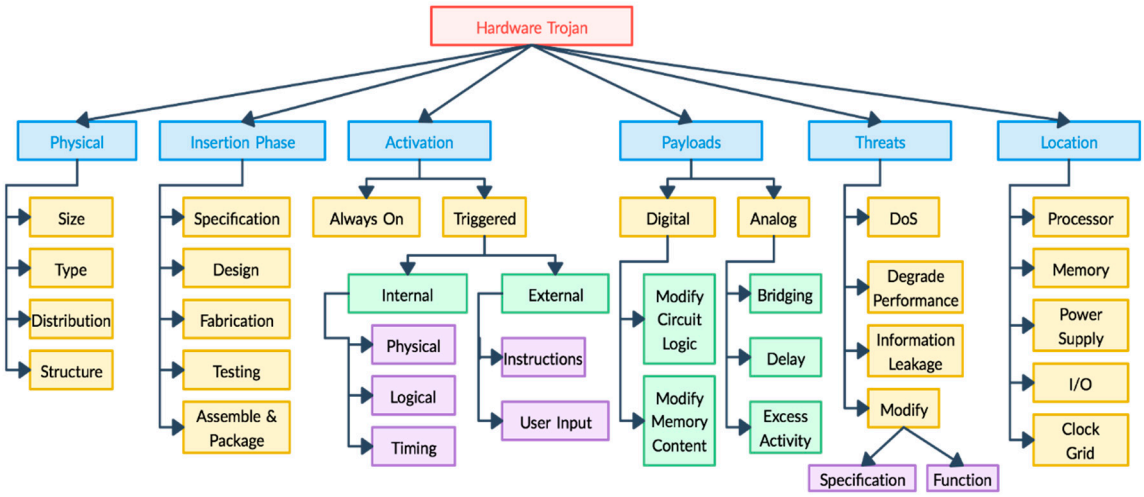

Figure 2. Taxonomy of hardware Trojan. DoS: Denial of Service; I/O: Input/Output.

Payload: A payload is the information that links a deterministic event with activation of the Trojan. Once the expected condition is detected by the trigger, the payload is activated, and Trojan starts performing the malicious activities. The payload mechanism of an HT is classified into digital and analog. Digital Trojans can modify logic values at specific internal payload nodes, or they can modify memory contents as well. Trojans with an analog payload can affect parameters of the circuit such as performance, power, noise, etc. Reference [18] shows certain examples of analog payloads, such as the addition of a bridging fault by inserting a resistor, increasing the capacitive load to affect the delay of the path, and the generation of excess activity in a circuit to affect its lifespan.

Threats: The classification of HTs can be done based on the threats they pose. HTs can lead to Denial of Service (DoS) attacks, performance degradation, and information leakage. Other threats that HT present are modification of function of the chip by adding or removing logic, or modification of specifications of the chip, which refers to alteration of the parametric properties of the chip such as delay [13].

Location: This refers to the physical location of the Trojan in the circuit. The locations where HTs can be present are processor, memory, power supple, Input/Output (I/O), and clock grid [19].

3.3. Hardware Trojan Insertion

Globalization and outsourcing of chip manufacturing open the gates for HT insertions at any point during the fabrication or design process, either in untrusted foundries or in design houses. Reference [20] discusses various Trojan injection phases during the IC lifecycle, as shown in Figure 3:

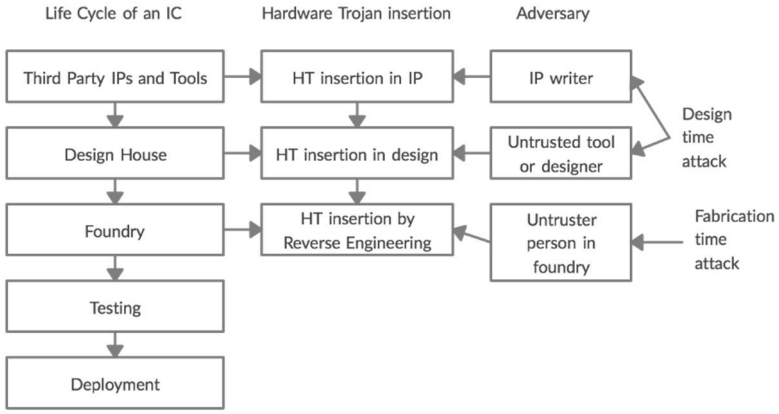

Figure 3. Hardware Trojan attacks during the IC lifecycle. IC: Integrated Circuit. HT: Hardware Trojan IP: Intellectual Property.

The modern business model of IC manufacture involves a design phase occurring at a separate facility using third-party IP (Intellectual Property) cores and design tools. Design-time attacks can occur at the design house or by using an HT-infected third-party IP core. The chip is transferred to a foundry for fabrication after the design phase where fabrication-time attacks can take place. Some sophisticated HT attacks, such as fault attacks in the advanced encryption system (AES) module, can circumvent both presilicon and postsilicon testing [20]. An example of a very sophisticated fabrication-time attack is the dopant-level HT. Dopant-level Trojans are very difficult to detect as they don't involve any addition or removal of existing gates and wires [21]. Reference [21] proposes a new fabrication-time attack that effectively reduces the system's security and makes a successful privilege escalation attack.

Being stealthy is one of the main characteristics of HTs to escape the detection during postsilicon validation. To achieve this, the attackers design combinational or sequential Trojans that are triggered by specific rare events. A sequence of rare events over a specific period of time triggers the sequential Trojans, making them similar to a time-bomb. However, a combination Trojan is triggered after a rare combination of logic values of the internal nodes [22].

Reference [22] shows how shrewd attackers can circumvent design-hardening approaches like Ring Oscillator Network (RON) by reverse engineering and mounting Trojans while bypassing embedded ROs. For example, instead of adding a gate, a simple addition of a transistor to pulldown the payload node would have almost no diffusion capacitance load and no impact on delay path, as shown in Figure 4 below:

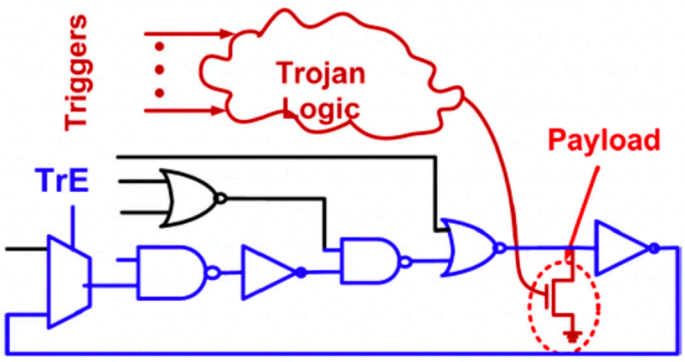

Figure 4. Payload insertion by-passing transistor payload [22].

HT attacks can be made in embedded memory, Static Random Access Memory (SRAM), which are considered to be design faults. SRAM array is an essential part of modern processors. These HT in SRAM array lead to no silicon-area overhead and may evade standard post-manufacturing testing.

The HT may lead to cell failure when inserted in the SRAM cell, or change the contents stored in the cell. The paper uses the Figure 5 below to demonstrate an HT attack in an SRAM array.

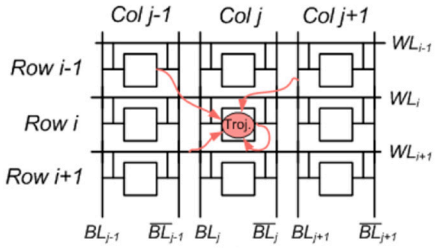

Figure 5. Hardware Trojan insertion in SRAM (Static Random Access Memory) array [22].

Reference [23] presents different attack techniques using HTs designed at Register Transfer Level (RTL) level, i.e., with modified Alpha Hardware Description Language (HDL) code. We have summarized various Trojan Types in Table 1 based on their trigger mechanism, area overheard, and detection possibility. The table shows eight Trojan types, with area overhead ranging from −9.4% to +6.8%. All the Trojans have different trigger mechanisms, such as Trojan type 2 gets triggered when the "F12" key is pressed. The detection possibility gives the possibility of Trojans being detected by functional testing and power trace testing. Trojan types 1, 3, 4, and 6 are inserted where the Finite State Machine (FSM) is inserted in the alphatop.v module. Trojan type 2 code is inserted where the "F12" key is defined in the kb2ascii.v file. Type 5 is located in the async_transmitter.v module. For Trojan type 7, one async_receiver.v module is added along with two control signals: tro7_trigger_rst and tro7_trigger_tx. Trojan code 8 is located in kb2ascii.v and kbtop.v modules.

Table 1. Trojan types at RTL (Register Transfer Level) level with area overhead and detection possibility.

Trojan Type	Trigger Mechanism	Area Overhead (Flip-Flops & Lookup Tables (LUTs))	Detection Possibility		Insertion Module
			Functional	Power	
1	Keyboard input "New Haven"	+0.8% & +6.8%	Probably not	Possible	Alphatop.v
2	"F12" key pressed	−9.4% & +0.024%	Probably not	Probably not	Kb2ascii.v
3	Word "Moscow" detected in plaintext	+3.3% & +2.4%	Probably not	Possible	Alphatop.v
4	Overflow of input buffer	+0.068% & +1.8%	Probably not	Possible	Alphatop.v
5	Change of key index	+0.75% & +1.4%	Almost impossible	Probably not	Async_transmitter
6	Inserted counter increment until it exceeds a predefined number	+0.34% & +0.17%	Almost impossible	Probably not	Alphatop.v
7	Unused RxD port	−4.4% & +4.9%	Almost impossible	Probably not	Async_Receiver.v
8	"Caps Lock" or another undefined key	−5.3% & +2.6%	Almost impossible	Probably not	Kb2ascii.v & kbtop.v

Reference [24] describes HT attack models in detail. The model differentiates different HT attacks on the basis of the different adversaries involved. Table 2 indicates each model, with the untrusted party involved in the model. The models consider three parties involved in the supply chain: third-party IP (3PIP) vendor, foundry, and SoC developer. The models are described as follows:

- Model A: SoC developers cannot develop all necessary IPs in house, so they purchase third-party IP (3PIP) cores, which could contain Trojan. Untrusted third-party IP vendors are adversaries in this case.

- Model B: Untrusted design houses are adversaries in this model. Fabrication is outsourced by fabless design house to third-party foundries with advanced process technologies. An adversary can insert an HT in the foundry.
- Model C: Third-party Electronic Design Automation (EDA) tools or fraudster designers are adversaries in model C. These tools and engineers or designers are involved in the process to fulfil the demands created by the increased complexity of SoC designs.
- Model D: Commercial Off-The-Shelf (COTS) products are the products that don't require custom development and are available off the shelf. They are cheaper but not trustworthy. None of the development stages are trusted in this model.
- Model E: This model considers all of the supply chain as a potential adversary except the foundry. The product could be designed or developed in an unfriendly foreign country, or a Trojan may be inserted by a counterfeiter into the original design after cloning the IC.
- Model F: This model assumes the entire supply chain to be adversary except the SoC developer. It combines Model A and B. Some companies integrate third-party IP cores into their SoC designs, and third-party foundries fabricate the chips. Only the SoC developer is trusted.
- Model G: This model applies to companies who outsource both application-specific integrated circuit (ASIC) design and fabrication. Therefore, both the system design integrator and foundry cannot be trusted. Clients receive the chips after they are fabricated, tested, and packaged.

Table 2. Trojan Models based on untrusted parties.

Model Number	Trojan Model	Untrusted Parties
A	Untrusted 3PIP (Third-party IP)	3PIP vendor
B	Untrusted fabless design house	Foundry
C	Untrusted SoC (System on a Chip) developer	SoC developer
D	Untrusted commercial off-the shelf (COTS)	3PIP vendor, foundry and SoC developer
E	Untrusted design	3PIP vendor and SoC developer
F	Untrusted outsourcer	3PIP vendor and foundry
G	Untrusted system integrator & foundry	System integrator and foundry

3.4. Side-Channel Attacks

Side-channel attacks (SCA) are noninvasive hardware-based attacks in which adversaries use forensic techniques related to the physical implementation of the embedded hardware to extract information [25]. SCAs pose a real threat to embedded devices as they can extract private hidden information from a machine by monitoring and measuring power use, electromagnetic emissions during cryptographic operations, or tendencies and frequencies of the computer to establish patterns. Side-channel information such as power, electromagnetic radiations, timing information, or even sound can be analyzed to implement this attack. This information gathered from a system without accessing the system directly coupled with appropriate calculations can help the attacker retrieve the secret cryptographic key. Reference [26] proposes a wireless interceptive SCA technique by performing a Correlation Electromagnetic Analysis (CEMA) attack against an AES-128 encryption algorithm for IoT applications, and successfully reveals the secret key.

Side-channel attacks are often classified into two categories: active side-channel attacks and passive side-channel attacks. Active attacks involve exploitation of the side-channel inputs where the device's functioning is tampered, such as fault-injection attacks in which error is induced in the computation. Passive side-channel attacks are the ones that exploit side-channel output, where an adversary simply observes the behavior of the device without disturbing it, such as in an electromagnetic analysis attack, which is described later [27].

Attackers require special and expensive equipment such as probes, an oscilloscope, a bandwidth amplifier, analyzing software, etc. Nonetheless, invention of the new powerful low-cost processors to be used in IoTs are making low-cost automated tools more accessible to the hackers. Some of the techniques to perform a side-channel attack are discussed as follows:

Acoustic cryptanalysis key extraction attack: Reference [28] demonstrates how such an attack can extract full 4096-bit RSA decryption keys from laptop computers (of various models), within an hour, using the sound generated by the computer during the decryption of some chosen ciphertexts. The research experimentally determined that such attacks could be carried out using either a plain mobile phone placed next to the computer or a more sensitive microphone placed 4 m away.

Differential Fault Analysis/Attack (DFA): This attack is well known for attacking symmetric block ciphers. A fault is inserted in the last round of cipher and is used to observe differences between correct and faulty ciphertexts [29].

Power Monitoring Attacks: In such attacks the adversary analyzes varying power consumption by a hardware device. These attacks have the capability of extracting cryptographic keys from the device. Simple power analysis (SPA) is straightforward and uses visual interpretation of power traces. However, differential power analysis (DPA) is a more advanced attack that is based on an analysis of measurements of power levels at different parts of the chip in addition to statistical analysis to overcome countermeasures such as added noise. The analysis helps the adversary identify computational operations being done by a device and reveals several bits of the crypto-key at a time. Full key is derived after a few repetitions of the whole process [30].

Electromagnetic Analysis Attacks (EMA): This attack is based on the analysis of captured electromagnetic radiation from a hardware device. These emanations could be captured from a distance and used to reconstruct the information being displayed on a computer monitor. These attacks have also been performed on integrated chips (IC) by placing tiny antennas close to the victim IC. Reference [9] discusses EMA attack and shows an image with a computer display reconstructed from electromagnetic radiations of a computer screen.

3.5. Hardware Trojan Based Side-Channel Attacks

Some HTs are designed and inserted to instigate side-channel attacks when triggered. Reference [31] proposes a new class of HT, which generates artificial side channels to leak information in a previously considered side-channel-resistant environment. These HTs are referred to as "Trojan side channels" (TSCs), and emphasis is made on using less than 100 gates to design them. The lightweight implementation of TSCs evades detection during conventional testing. Another reason of difficult detection of these HTs is that they do not affect the device's functionality in any way. The first TSC design is based on code-division multiple access (CDMA) communications. In CDMA, code bits change much faster than information bits, as CDMA uses many code bits to transfer single information bits. The TSC design creates a CDMA code sequence by using a pseudo-random number generator (PRNG). This sequence is then used to modulate information bits using XOR gate. Then, a covert CDMA channel in the power-side channel is set up by forwarding the modulated sequence to the leakage circuit (LC). For encoded one bits, all zero-to-one transitions of the code in the circuit have higher leakage, whereas, for all encoded zero bits, one-to-zero transitions of the code have higher leakage. These two different code sequences are used by the adversary for demodulation. The actual transferred bit is identified by the higher correlation coefficient. The second TSC design demonstrates an attack on the AES-128 block cipher and its key schedule. Leaking intermediate states are introduced in the key schedule artificially using known input bits and key bits. Then, a differential power analysis attack is used to discriminate between wrong and right key bits using the correlation coefficient.

Reference [32] demonstrates how easy it is to insert an HT which induces side-channel leakage to recover a key once triggered. This Trojan is inserted by modifying a few gates during manufacturing to increase path delay. Detecting such an HT is very difficult as inserting it does not require the addition or removal of any logic. The HT is inserted in the ASIC platform by modifying a few transistors at

sub-transistor level to increase their path delay. In case of the FPGA platform, the HT is inserted to delay the path of Look-Up Tables (LUT) by lengthening the selected signals through switch boxes, or by inserting a route-through LUT. Then, the clock frequency is increased beyond its maximum, where SCA leakage becomes exploitable and the Trojan starts leaking information through side channels.

Reference [33] presents HT-based algebraic fault analysis (AFA) of a lightweight block cipher HIGHT. HIGHT is one of the lightweight encryption algorithms that is considered for implementation in an IoT. The HT is assumed to be inserted in Register Transfer Level (RTL) by an adversary and introduces a fault by flipping a specific bit of a certain intermediate word of the cipher when it is triggered. A merged equation system for the cipher and faults is constructed, and the secret key is recovered by solving the equation system using an automatic satisfiability based (SAT-based) solver.

4. Countermeasures

4.1. Trojan Detection

Security must be entrenched in the system at every single level from design until its actively running, for its entire lifetime. A common way to address HT attacks is Trojan detection. Launching a successful attack without being detected is usually one of the main goals of the adversaries. "Previous research has identified that evading detection is a critical property for hardware Trojans designers." [21]. Existing designs and fabricated ICs are tested to verify that there are no modifications in their circuitry. IC designs are validated at the design stage, which is referred to as presilicon stage, and fabricated ICs are verified at the manufacturing stage, referred to as postsilicon stage [24]. We classify Trojan Detection into presilicon and postsilicon detection, as shown in Figure 6. Postsilicon is further classified into destructive and nondestructive detection techniques. A further description of each technique under presilicon and postsilicon is as follows:

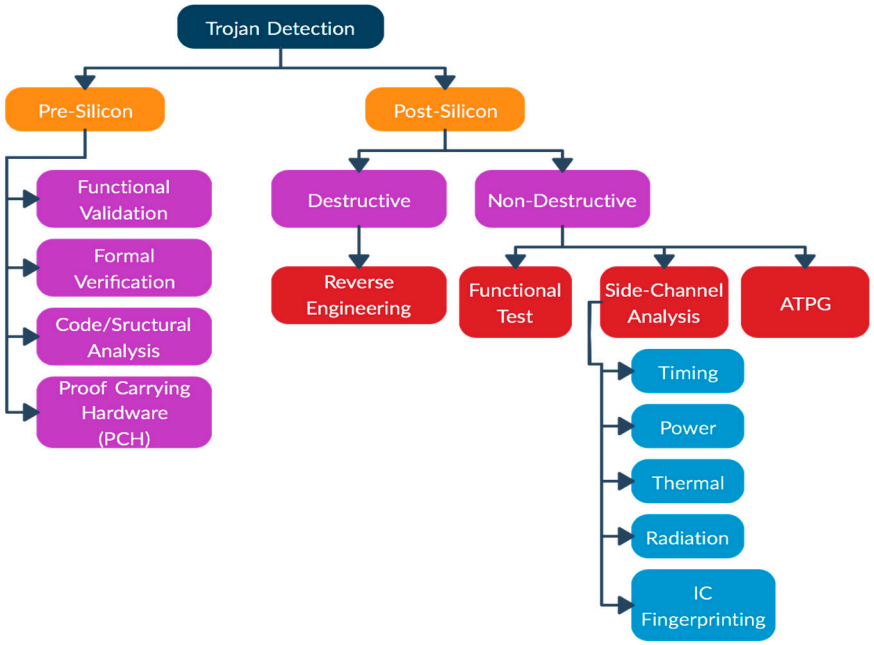

Figure 6. Hardware Trojan Detection Techniques. ATPG: automatic test-pattern generation.

Postsilicon Trojan detection techniques: These techniques can be classified into destructive techniques and nondestructive techniques.

Destructive Techniques: In destructive techniques, reverse-engineering is used, which involves depackaging an IC, and design-for-trust validation of the product is reconstructed from the obtained images of each layer. This method provides 100% detection rate of any malicious modification in

the IC, but it is very high-cost and time consuming. Also, the chip cannot be used after the process, and information of only that particular chip is gained. This method is good to perform on a few samples to obtain the golden IC model's characteristics. Golden chip is a trojan-free, perfectly functioning IC that is used to be matched against other ICs [24].

Nondestructive Technique: These techniques are used to verify fabricated ICs from untrusted foundry using the following methods:

- *Functional test*: These tests require the activation of Trojans by using test vectors, and then the responses are compared with the correct results. The stealthy trojans gone undetected during manufacturing test process can be detected during these tests. The Trojans that do not change the functionality of the original circuit may not get detected by the functional tests. References [12,24] discusses that in order to hide from accidental triggering and side-channel analysis, HTs try to choose inactive nets (nets that rarely switch) in order to keep power leakage or extreme nets (nets that spend most time on one state) toa minimum. Circuit-under-test (CUT) can be evaluated, and inactive nets can be found by calculating the probability of switching of all the nets. Reference [12] proposes a method to detect low active nets on a CUT on function mode.
- *Side-channel analysis*: This method takes advantage of side effects produced by additional circuits or Trojan activation, i.e., extra path delay, power, heat, etc., which are measured to detect HTs [24]. An IC fingerprinting technique uses side-channel information such as power, temperature, EM, etc., to construct fingerprints with noise modeling for an IC family. Reference [34] demonstrates a research that was able to distinguish a genuine IC from the one with a Trojan down to 0.01% of the size of the main circuit using IC fingerprinting. Most of these techniques require comparison against golden ICs, which is not always available. Also, highly sensitive instruments are needed to measure small side-channel signals such as leakage current, as Trojans require very little current due to their size. References [14,35] present an HT detection method based on side-channel analysis which shows that measuring path delay on 20 paths can help detect more than 80% of HTs.
- *ATPG (automatic test-pattern generation) method*: This method uses the application of digital stimulus to the chip and analyzes the digital output. Reference [36] discusses a few researches made based in the automatic test-pattern generation (ATPG) tool to detect HT. One scheme called MERO (Multiple Excitation of Rare Occurrence) uses a statistical approach based on generating test vectors, which excite rare nodes simultaneously and increase the probability of an HT being triggered and detected easily. The research proposes the use of this technique with side-channel detection techniques to increase their impact. Another ATPG-based research discussed in [36] changes the design rules to detect HT that is inserted in the chip's existing logic.

Presilicon Trojan detection technique: SoC developers and design engineers use these techniques to validate 3PIP cores and their designs. The presilicon detection techniques can be classified into the following three classes:

- *Functional validation*: This technique uses the same principle notion mentioned in the functional test described under postsilicon techniques. Functional tests are performed on a tester, which involves collecting output responses to each input pattern provided, but functional validation is conducted with simulation using existing functional testing techniques [24].
- *Code/Structural analysis*: Behavioral or structural codes are tested to detect any redundant statements or circuits that might be associated with a Trojan. These techniques do not offer detection guarantee and might require manual postprocessing diagnosis of suspicious signals or gates [24].
- *Formal verification*: This approach requires logic verification of target design that should satisfy a predefined set of security properties. This technique might not detect other unexpected functionalities caused by a Trojan beyond these predefined security properties [24].
- *Proof Carrying Hardware (PCH)*: A PCH framework uses an interactive theorem prover to verify security properties on soft IP cores. Soft IP cores are synthesizable cores and exist as a netlist (a list

of logic gates and interconnections used in an integrated circuit) or as a hardware description language (HDL) code. Reference [37] presents a theorem-proving and equivalence-checking approach using PCH to protect IP cores and verify the absence of malicious implants in the IP blocks.

The majority of the detection techniques require expensive and sophisticated tools. Reference [38] proposes an inexpensive method to create and detect HTs in dormant as well as active state using off-the-shelf technologies.

4.2. Design for Trust

None of the existing HT detection techniques provide a complete detection guarantee. Therefore, embedding HT prevention methods during the design phase through design-for-trust provides a more effective potential approach as a countermeasure against HTs. Design-for-trust (DfT) involves the prevention of HT insertion, facilitating the detection of HTs using approaches discussed in the Trojan detection section and trustworthy computing on untrusted components.

Facilitating detection involves the use of techniques to increase the sensitivity and probability of HT detection during functional tests and side-channel analysis. The ATPG-based scheme presented in [36] is an example of such techniques that increase the HT detection probability. Another method to facilitate detection is runtime monitoring.

Runtime Monitoring: Although many techniques have been developed to detect HTs during presilicon and postsilicon tests, it is very difficult to trigger and detect all types of Trojans. Runtime monitoring monitors the chip's behavior or operating conditions to detect HTs during the chip's run-time. On detecting an abnormality during run-time, they can disable the chip, bypass it, or trigger other security measures to reduce the effects of HTs and provide a reliable operation. The approach is appealing to detect HTs but comes with some performance overhead. Runtime monitoring can be classified into three classes: configurable security monitors, hardware7ysoftware approach and variant-based parallel execution. Runtime monitoring can be used as a complementary approach with chip-detection testing phases to detect HTs [24]. The classification of these monitoring approaches is described as follows:

- *Configurable Security Monitors*: This approach uses security monitors to facilitate real-time functionality monitoring by adding reconfigurable logic in an SoC. The signal behavior is checked by feeding the signals to Security Monitors (SMs). The configuration of the SM is done to implement FSM, and it does not disturb the normal operation of the system. When an abnormality is detected, the checking occurs simultaneously with the other operations of the system and triggers countermeasures as needed. To perform different security checks, SMs can be reconfigured as well [22].
- *Variant-Based Parallel Execution*: This approach performs the simultaneous execution of multiple functionally equivalent variants on different processing elements (PEs), and the results are compared. On detecting a mismatch, a new PE is involved until a match is possible and Trojan-infected PEs are identified [24]. The more efficient the generation of variants, the more effective is the process. This approach can help computers with multi-core processors achieve a high level of trust, but with an additional cost of computational, performance, and energy overhead.
- *Hardware–Software Hybrid Approach*: Trojan detection and tolerance by a software solution can provide efficient safety for systems with microprocessors [22]. This approach uses design-verification tests to identify unused circuitry and marks it suspicious. This technique circumvents HTs by removing the suspicious circuitry and replacing it with exception logic [24]. A software exception is triggered which allows the system to bypass the HT and operate normally [22].

Prevention of HT Insertion

The next part of DfT is prevention of HT insertion. The attackers need to understand circuit functionality before they are able to insert Trojans, and they usually use reverse engineering to do so. Obfuscation, layout filler, camouflaging, and use of trust modules are some techniques to thwart the insertion of HTs.

- *Obfuscation*: This approach is based on making HT insertion difficult for an attacker by obscuring functionality and structural properties of a design. The obfuscation approach is based on applying a key-based obfuscation technique. This technique modifies a circuit's state transition function, which makes it operate in two modes, i.e., a normal mode and an obfuscated mode. An obfuscated mode produces incorrect functional behavior, generating incorrect output, while the normal mode produces desired output. Thus, internal circuit nodes are obscured, and it makes it difficult to identify genuine functionality. This approach thwarts the ability of an adversary to insert Trojans without knowing the circuit functionality and right input vectors. Also, a Trojan might be active only in obfuscated mode, which makes it benign. The right function of the circuit appears only when the right key is applied. Logic obfuscation is classified into combinational and sequential logic obfuscation. In combinational logic obfuscation, the addition of XOR or XNOR gates in design is performed randomly. The key-gate is provided one functional input and one bit key input. Tamper-proof memory inside the design is used to store the correct key. For sequential logic obfuscation, the functional state of a finite-state machine is concealed by introducing additional states and transformations [20,24,39].

Camouflaging is another obfuscation technique done at the layout level. In a camouflaged logic gate, fake connections are added between the layers to create indistinguishable layouts. This technique of adding dummy contacts prevents the attackers from obtaining a correct gate-level netlist of a circuit from the layout. Thus, hindering the attacker's ability to insert an HT in the original design [24].

- *Layout Filler*: One of the common ways that attackers use to insert an HT in a circuit is by placing the Trojan in unused vacant spaces in the circuit. To prevent HT insertion, the empty spaces in the circuit layout are filled with functional filler cells using a built-in self-authentication (BISA) approach. The reason why filler cells are functional is so that the attacker cannot replace these filler cells with a Trojan without changing the functionality of the circuit. The functional filler cells form a combinational circuitry. Any abnormal function detected during testing indicates the replacement of filler cells with a Trojan [7,24].

The DfT techniques discussed above are still difficult to implement, as most of them require adding or modifying circuitry, which affects the performance of the chip, increases the complexity of processing, adds area/time overhead, and might increase power leakage [24].

4.3. Split Manufacturing

Another method to minimize risks to an IC design is the split manufacturing process. The objective of this approach is to enhance the security of an IC by hiding the design intent and preventing malicious insertion. The design is divided into Front End of Line (FEOL) and Back End of Line (BEOL) parts by different foundries for fabrication. The FEOL layers, i.e., transistors and low-metal layers, are fabricated in an untrusted foundry using an advanced process. The untrusted foundry then ships wafers to a trusted foundry. The trusted foundry uses a less advanced process to fabricate back end of line layers, i.e., high-level interconnects. The untrusted foundry cannot find places in a circuit for Trojan insertion as it does not have access to the BEOL layers [20,24,39].

4.4. Other Hardware Security Techniques for IoTs

A combination of various techniques is usually a better approach towards implementing security, however it depends on the level of security appropriate for each individual device.

- *Device Identity*: Device identity is necessary to enforce accountability. Organizations are using Public Key Infrastructure (PKI) for securing their digital communication with digital certificates. Digital certificates are used to implement various security controls such as device authentications, securing host-to-host communication, and securing communications using TLS/SSL communication protocols. PKI's ability to scale makes it a great infrastructure for device identity and authentication. However, due to the limited lifespan of digital certificates, the device manufacturer must consider the lifespan of these certificates and combine PKI with a proper certificate management method [40].
- *Hardware Security Module (HSM)*: IoT devices that are easily accessible are prone to physical attacks, which is why there is a need to use tamper-protected hardware security modules to protect information such as cryptographic key and operations such as data encryption, PIN verification, etc. HSM is a physical device used to provide an extra layer of security around cryptographic keys, trade secrets, and other sensitive applications or data. The HSM device can be embedded in another hardware, connected to a server, or be used as a standalone device. Giving each device a unique electronic identity would increase the authenticity of the device, and this is achievable by injecting a semiconductor chip with a unique identity in each device. This process is called key injection. The integrity of the key injection process is of extreme importance and can be achieved by using HSMs as they establish a Root of Trust for this process. HSMs are capable of creating, securing, and managing these keys [41].
- *Trusted Platform Module (TPM)*: Trusted Computing Platform Alliance (TCPA) came up with TPM as hardware instantiation of TCPA specification with a focus on ensuring privacy and enhancing security. TPM is a secure microprocessor with added cryptographic functionalities and comes in various shapes and sizes: as discrete or embedded hardware devices, or firmware implementations. TPM hardware provides platform root of trust and is capable of extending its trust to other parts of the platform by building a chain of trust. TPM has a built-in RSA engine, that can perform up to 2048-bit RSA encryption/decryption, which is used when it does digital signing and key wrapping. TPM contains a built-in hash function as well to compute hash values of small pieces of data. TPM uses a set of special-purpose Platform Configuration Registers (PCR), which store single cryptographic hash and are readable by external software. Each binary in the boot chain computes a hash of the subsequent binary and extends it into the PCRs, as well as records it in measurement log. The sequence of these measured values can be compared against current PCR values to check for integrity of the measurement log. TPM also provides attestation by signing the PCR values with its attestation key. Attestation Identity Keys (AIK) are used to provide platform authentication. Certificates available within TPM are used to create AIKs [42]. TPM may hold three types of certificates:

  (1) Endorsement certificate that contains a public key of EK, which stands for Endorsement Key (public/private key pair unique to every TPM) to provide attestation about the legitimacy of a TPM.
  (2) The Platform certificate that provides attestation of a platform's security components.
  (3) The Conformance certificate that provides attestation by an accredited party of platform's security properties [42].

TPM provides identity with the use of EK and AIK. TPM also provides the ability to attest to the software and allows authorizations based on software identity. The Trusted Execution Environment (TEE) in TPM assures functioning of operations without modification and mitigates both hardware and software attacks. The challenge of using TPM in IoT devices is the additional space and power requirement, as well as the increased cost for such resource-constrained devices [42].

- *Roots of Trust (RoT)*: Roots of trust in a computing system are hardware/software components that are inherently trusted by the computer's OS. It is a source that a cryptographic system can always trust. It must be implemented in hardware ideally and is trusted to protect cryptographic

keys, perform device authentication, or verify software. RoT may include a variety of components such as security perimeter to define the requirements for SoC protection, secure CPU, runtime memory to protect runtime data, tamper resistance, True Random Number Generator (TRNG), secure clock, secure counter, and secure storage [43]. RoT provides the functionalities behind trusted computing features such as drive encryption, storage protection, secure communication, key management, and secure monitoring, where hardware Root of Trust would notify the host about an attempt to add malicious insertion that has been made. Root of Trust is critical to create an embedded HSM for an IoT [43].

- *Physical Unclonable Function (PUF)*: PUF is a lightweight security primitive that can be exploited for device identification, authentication, secret key generation, and storage. They are unique due to their unclonability and tamper-evident properties. PUFs are similar to one-way functions as the mapping between inputs and outputs is repeatable but also irreversible. "Challenge" is the electrical stimulus provided to a PUF which leads to a reaction called "response". The set of challenge-response pairs (CRP) characterizes a PUF instance and represents a unique feature used for its identification. PUFs are considered to be tamper-evident, as any malicious modifications of the PUF structure are easily detectable. Reference [44] proposes a method for a hardware mutual authentication protocol using PUF. Reference [45] proposes a way to implement a secure communication protocol for an IoT based on PUF. Traditional security mechanisms used EEPROM (electrically erasable programmable read-only memory) and battery-backed nonvolatile SRAM in order to store secret keys, which are vulnerable to various attacks. Therefore, the use of tamper-resistant devices to store and defend keys from attacks is crucial. The resource limitation in IoT devices makes it difficult to use traditional methods to store these keys. Silicon PUFs provide a promising security alternative for IoT devices. The key generated by PUF is tamper resistant, as physical tampering damages underlying nanoscale structural disorder. Ring oscillator PUFs are a great choice for key generation as they produce a limited number of CRPs for authentication, which makes it very desirable for small-sized devices [46].
- *Device Identifier Composition Engine (DICE)*: DICE is a security standard created by Trusted Computing Group (TCG) for hardware-based cryptographic device identity, attestation of device firmware, safe deployment, and data encryption. Its modest hardware requirements make it perfect for use in resource-constraint IoT devices for security and privacy and it can be implemented in hardware of security products during manufacturing. It uses a one-way hash function simple equation and can easily be implemented in a tiny microcontroller. DICE architecture is designed to address the need for security enhancement in IoT devices, especially where Trusted Platform Modules (TPM) may be unfeasible due to limited resources. It works by organizing the boot into layers. A unique "secret" is computed by each layer which is kept confidential by each layer. The configuration is based on Unique Device Secret (UDS). Whenever a different code is booted, the secrets change. If a secret is disclosed due to a vulnerability, the device is automatically re-keyed, and secrets are protected [47]. Microsoft was the first to adopt DICE with Azure IoTs and new Device Provisioning Services (DPS) [48].

5. Conclusions

In conclusion, IoT devices cannot be completely secure until we lay a strong foundation of secure hardware for them. All the efforts and cost made in order to secure these devices could go to waste if the device has an HT inserted which can get triggered anytime and destroy the whole device. Embedded hardware security in IoTs is a necessity to protect the "identity" of devices to protect them against tampering, and to protect the privacy and security of data they generate. Use of TPM, DICE, and other methods discussed above provide another layer of security around the cryptographic keys that are used to protect the integrity, confidentiality, and authenticity of the system and data. A smart hardware security should be designed not only to protect the device and data against attacks, but also to react in the best way possible and maintain privacy during attack or destruction. Security

should not be an afterthought. The IoT market needs proper security standards in place, especially for the design phase, as most of the techniques focus on security in the postsilicon phase. Securing the environment and introducing good policies for secure manufacture of the chips while keeping the privacy of third-party companies in mind is a necessity to achieve a high level of hardware security. The paper describes the HT in detail as it is the biggest threat to the hardware security along with the role that side-channel analysis plays in it. The taxonomy of the HT is presented in detail in the paper. Understanding HT taxonomy can help researchers find new techniques to detect and deter each kind of HT on the basis of its classification. The comprehension of different HT insertion stages throughout an IC's lifecycle is another factor worth researchers' attention, to determine and prevent HT insertion in these various vulnerable stages. Also, countermeasures are presented, which provide detection techniques, design for trust, split manufacturing for trust, and other techniques such as HSM and TPM to enhance hardware security.

Author Contributions: S.S. initiated and conducted the research, and wrote the paper. B.J.M. was responsible for guiding the hardware security part, helped in the security analysis and writing the paper. T.H. helped in organizing the work, supervising and guiding the project, and revising the paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tao, H.; Bhuiyan, Z.; Abdalla, A.; Hassan, M.; Zain, J.; Hayajneh, T. Secured Data Collection with Hardware-Based Ciphers for IoT-Based Healthcare. *IEEE Internet Things J.* **2019**, *6*, 410–420. [CrossRef]
2. Health IT Security. Available online: https://healthitsecurity.com/news/healthcare-ransomware-medical-device-security-key-2018-trends (accessed on 9 March 2019).
3. Maple, C. Security and privacy in the internet of things. *J. Cyber Policy Issue 2 Internet Things* **2017**, *2*, 155–184. [CrossRef]
4. Fremantle, P.; Scott, P. A security survey of middleware for the Internet of Things. *Peer J.* **2015**, *3*, e1521.
5. Hayajneh, T.; Bassam, J.M. Lightweight Block Ciphers for IoT: Energy Optimization and Survivability Techniques. *IEEE Access* **2018**, *6*, 35966–35978.
6. Bassam, J.M.; Hayajneh, T.; Vasilakod, A.V. A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues. *J. Netw. Comput. Appl.* **2015**, *58*, 73–93.
7. Bassam, J.M.; Hayajneh, T.; Khalaf, Z.A.; Yousef, K. Modeling and optimization of the lightweight HIGHT block cipher design with FPGA implementation. *Secur. Commun. Netw.* **2016**, *9*, 2200–2216.
8. Hayajneh, T.; Bassam, J.M.; Yousef, K.; Khalaf, Z.A.; Bhuiyan, Z. Hardware Design and Modeling of Lightweight Block Ciphers for Secure Communications. *Future Gener. Comput. Syst.* **2017**, *83*, 510–521.
9. Gusmeroli, S.; Piccione, S.; Rotondi, D. IoT Access Control Issues: A Capability Based Approach. In Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012), Palermo, Italy, 4–6 July 2012; pp. 787–792.
10. University of Cincinnati. Available online: http://gauss.ececs.uc.edu/Courses/c6056/pdf/hardware.pdf (accessed on 4 March 2019).
11. Rostami, M.; Koushanfar, F.; Karri, R. A Primer on Hardware Security: Models, Methods, and Metrics. *Proc. IEEE* **2014**, *102*, 1283–1295. [CrossRef]
12. Zou, M.; Cui, X.; Shu, L.; Wu, K. Potential Trigger Detection for Hardware Trojans. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 1384–1395. [CrossRef]
13. Venugopalan, V.; Patterson, C. Surveying the Hardware Trojan Threat Landscape for the Internet-of-Things. *J. Hardw. Syst. Secur.* **2018**, *2*, 131–141. [CrossRef]
14. Wang, X.; Salmani, H.; Tehranipoor, M.; Plusquellic, J. Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis. In Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, Boston, MA, USA, 1–3 October 2008; pp. 87–95.
15. Kounelis, F.; Sklavos, N.; Kitsos, P. Run-Time Effect by Inserting Hardware Trojans, in Combinational Circuits. In Proceedings of the Euromicro Conference on Digital System Design (DSD), Vienna, Austria, 30 August–1 September 2017; pp. 287–290.

16. Koley, S.; Ghosal, P. Addressing Hardware Security Challenges in Internet of Things: Recent Trends and Possible Solutions. In Proceedings of the 12th IEEE International Conference on Advanced and Trusted Computing (UIC-ATC-ScalCom-CBDCom-IoP), Beijing, China, 10–14 August 2015; pp. 517–520.
17. Wang, X.; Tehranipoor, M.; Plusquellic, J. Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions. In Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 9 June 2008; pp. 15–19.
18. Bhunia, S.; Narasimhan, S.; Chakraborty, R. Hardware Trojan: Threats and emerging solutions. In Proceedings of the 2009 IEEE International High Level Design Validation and Test Workshop, San Francisco, CA, USA, 4–6 November 2009; pp. 166–171.
19. Kumar, S.; Mahapatra, K. How to Protect Your Device from Hardware Trojans. In Proceedings of the Real World IoT Security Conference, Bangalore, India, 20 June 2017.
20. Bhunia, S.; Hsiao, M.; Banga, M.; Narasimhan, S. Hardware Trojan Attacks: Threat Analysis and Countermeasures. *Proc. IEEE* **2014**, *102*, 1229–1247. [CrossRef]
21. Yang, K.; Hicks, M.; Dong, Q.; Austin, T.; Sylvester, D. A2: Analog Malicious Hardware. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 18–37.
22. Wang, X. Hardware Trojan Attacks: Threat Analysis and Low-Cost Countermeasures through Golden-Free Detection and Secure Design. Master's Thesis, Case Western Reserve University, Cleveland, OH, USA, 2014.
23. Jin, Y.; Kupp, N.; Makris, Y. Experiences in Hardware Trojan Design and Implementation. In Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), San Francisco, CA, USA, 27 July 2009; pp. 50–57.
24. Xiao, K.; Forte, D.; Jin, Y.; Karii, R.; Bhunia, S.; Tehranipoor, M. Hardware Trojans: Lessons Learned after One Decade of Research. *J. ACM Trans. Des. Autom. Electron. Syst. (TODAES)* **2016**, *22*, 6. [CrossRef]
25. Chip Design. Available online: http://eecatalog.com/chipdesign/2018/10/04/iot-security-starts-with-threat-modeling-and-security-analysis/ (accessed on 4 March 2019).
26. Pammu, A.; Chong, K.; Ho, W.; Gwee, B. Interceptive Side Channel Attack on AES-128 Wireless Communications for IoT Applications. In Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Jeju, Korea, 25–28 October 2016; pp. 650–653.
27. Spreitzer, R.; Moonsamy, V.; Korak, T.; Mangard, S. Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 465–488. [CrossRef]
28. Genkin, D.; Shamir, A.; Tromer, E. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 3552, pp. 444–461.
29. Breier, J.; He, W. Multiple Fault Attack on PRESENT with a Hardware Trojan Implementation in FPGA. In Proceedings of the IEEE International Workshop on Secure Internet of Things (SIoT), Vienna, Austria, 21–25 September 2015; pp. 58–64.
30. Search Security. Available online: https://searchsecurity.techtarget.com/definition/differential-power-analysis-DPA (accessed on 2 March 2019).
31. Lin, L.; Kasper, M.; Guneysu, T.; Paar, C.; Burleson, W. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems CHES, Lausanne, Switzerland, 6–9 September 2009; pp. 382–395.
32. Ender, M.; Ghandali, S.; Moradi, A.; Paar, C. The First Thorough Side-Channel Hardware Trojan. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 755–780.
33. Chen, H.; Wang, T.; Zhang, F.; Zhao, X.; He, W.; Xu, L.; Ma, Y. Stealthy Hardware Trojan Based Algebraic Fault Analysis of HIGHT Block Cipher. *Secur. Commun. Netw.* **2017**. [CrossRef]
34. Agrawal, D.; Baktir, S.; Karakoyunlu, D.; Rohatgi, P.; Sunar, B. Trojan Detection using IC Fingerprinting. In Proceedings of the IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; pp. 296–310.
35. Amelian, A.; Borujeni, S. A Side-Channel Analysis for Hardware Trojan Detection Based on Path Delay Measurement. *J. Circuits Syst. Comput.* **2018**, *27*, 1850138. [CrossRef]
36. Ranjani, R.S.; Devi, M.N. Malicious Hardware Detection and Design for Trust: An Analysis. *Elektrotehniski Vestn.* **2017**, *84*, 7–16.
37. Guo, X.; Dutta, R.; Jin, Y. Pre-silicon security verification and validation: A formal perspective. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015.

38. Rooney, C.; Seeam, A.; Bellekens, X. Creation and Detection of Hardware Trojans Using Non-Invasive Off-The-Shelf Technologies. *Electronics* **2018**, *7*, 124. [CrossRef]
39. Li, H.; Liu, Q.; Zhang, J. A Survey of Hardware Trojan Threat and Defense. *Integr. J.* **2016**, *55*, 426–437. [CrossRef]
40. Allerin. Available online: https://www.allerin.com/blog/authentication-and-device-identification-in-iot-security (accessed on 9 April 2019).
41. Utimaco. Available online: https://hsm.utimaco.com/solutions/applications/key-injection/ (accessed on 15 April 2019).
42. Bajikar, S. *Trusted Platform Module (TPM) Based Security on Notebook PCs—White Paper*; Mobile Platforms Group Intel Corporation: Santa Clara, CA, USA, 2002.
43. Synopsys. Available online: https://www.synopsys.com/designware-ip/technical-bulletin/understanding-hardware-roots-of-trust-2017q4.html (accessed on 9 April 2019).
44. Barbareschi, M.; De Benedictis, A.; Mazzocca, N. A PUF-based hardware mutual authentication protocol. *J. Parallel Distrib. Comput.* **2018**, *119*, 107–120. [CrossRef]
45. Chatterjee, U.; Chakraborty, R.; Mukhopadhyay, D. A PUF-Based Secure Communication Protocol for IoT. *ACM Trans. Embed. Comput. Syst.* **2017**, *16*, 1–25. [CrossRef]
46. Zhang, J. Physical Unclonable Function-Based Key Sharing for IoT Security. Available online: https://arxiv.org/pdf/1807.10884.pdf (accessed on 15 April 2019).
47. Mattoon, D. DICE: Foundational Trust for IoT. In Proceedings of the Microsoft Flash Memory Summit, Santa Clara, CA, USA, 8–10 August 2017.
48. Electronic Design. Available online: https://www.electronicdesign.com/embedded-revolution/roundtable-qa-device-identity-composition-engine-dice (accessed on 15 April 2019).