Secure and Efficient Initialization and Authentication Protocols for SHIELD

Chenglu Jin and Marten van Dijk

University of Connecticut
chenglu.jin@uconn.edu, marten.van_dijk@uconn.edu

June 15, 2016

Abstract

With the globalization of semiconductor production, out-sourcing IC fabrication has become a trend in various aspects. This, however, introduces serious threats from the entire untrusted supply chain. To combat these threats, DARPA (Defense Advanced Research Projects Agency) proposed in 2014 the SHIELD (Supply Chain Hardware Integrity for Electronics Defense) program to design a secure hardware root-of-trust, called dielet, to be inserted into the host package of legitimately produced ICs. Dielets are RF powered and communicate with the outside world through their RF antennas. They have sensors which allow them to passively (without the need for power) record malicious events which can later be read out during an authentication protocol between the dielet and server with a smartphone as intermediary.

This paper introduces a general framework for the initialization and authentication protocols in SHIELD with different adversarial models based on formally-defined security games. We introduce a "try-and-check" attack against DARPA's example authentication protocol in their call for SHIELD proposals which nullifies the effectiveness of SHIELD's main goal of being able to detect and trace adversarial activities with significant probability. We introduce the first concrete initialization protocol and, compared to DARPA's example authentication protocol, introduce an improved authentication protocol which resists the try-and-check attack. The area overhead of our authentication and initialization protocols together is only 64-bit NVM, one 8-bit counter and a TRNG based on a single SRAM-cell together with corresponding control logic. Our findings and rigorous analysis are of utmost importance for the teams which received DARPA's funding for implementing SHIELD.

Contents

1 Introduction

Outsourcing IC (Integrated Circuit) fabrication has become mainstream in IC design, fabrication, testing and packaging. Even though outsourcing to a trustworthy manufacturing facility for legitimate IC fabrication and assembly is assumed by default, legitimately produced ICs still need to pass through the remainder of a supply chain which is not in one's own control and can therefore not be trusted. This opens a whole new range of serious threats including compromise of IP (Intellectual Property) Privacy, IC Overbuilding, Reverse Engineering, and Counterfeit ICs. Due to these attacks, semiconductor industry not only loses 4 billion dollars annually [1] but also untrusted (expired or malicious) hardware has become common in embedded systems.

In order to have a sound basis for trustworthy embedded systems one would ideally want to have an additional point of trust within the supply chain such that somehow tamper-evidence can be added to legitimately manufactured ICs (and supply chain attacks can be prevented or detected). A first approach is to be completely in control of IC packaging so that ICs will be equipped with trusted packages which are smart in that they are into some extent tamper-evident. In other words, if tampering happens, then the package will gather irreversible evidence of the tampering. This evidence can be seen or "read-out" later to allow verification of the authenticity and integrity of the IC inside the package.

DARPA (Defense Advanced Research Projects Agency) takes this approach to a new level: Their SHIELD (Supply Chain Hardware Integrity for Electronics Defense) program [2] proposes to embed/insert an ineradicable hardware root-of-trust, called dielet, into the host package of every legitimately produced IC. The dielet is intelligent in that it is able to passively sense and record malicious behavior (such as unexpected exposure to light, vibration, etc.) and can be read-out at a later moment to gather any recorded tampering evidence. SHIELD goes beyond the simple first approach described above: Not the IC packaging process itself needs to be designed to offer tamper-evidence, one only needs to make sure that a process of inserting dielets into host packages of legitimately produced ICs is in place. This process of dielet insertion must be part of the trusted IC assembly since otherwise any (malicious) IC can be linked/bound to a valid dielet and later on pass identification and authentication as if the IC can be trusted to be what it claims to be through the dielet.

Notice that the proposed dielet technology transforms any host package into a tamper-evident one. Besides the smaller Trusted Computing Base (TCB) in the form of trusting the dielets and trusting the bond (= host package) between ICs and dielets (rather than trusting a larger overall tamper-evident packaging), SHIELD also provides the main advantage of backward compatibility: SHIELD technology is a sort of labeling technology which applies to already existing IC manufacturing and corresponding supply chains.

The main design features of a dielet are shown in Fig. 1, the entire authentication system contains three parts: a dielet inserted in the package of the host chip, a smartphone and a secure remote server. The remote server stores the information for identification and authentication, such as serial ID and cryptographic key for each dielet. The communication between the server and the dielet requires an inexpensive appliance to read the dielet. A smartphone with a probe as a common appliance can be used in practice. The communication channel between the server and the smartphone is over a wireless network and over the Internet while the dielet connects to the smartphone via an RF (Radio Frequency) channel. Because the dielet is passively powered up through an RF transceiver module, the dielet has to be *both area and energy efficient*.

The dielet provides a unique permanent identification and implements sensors that are capable

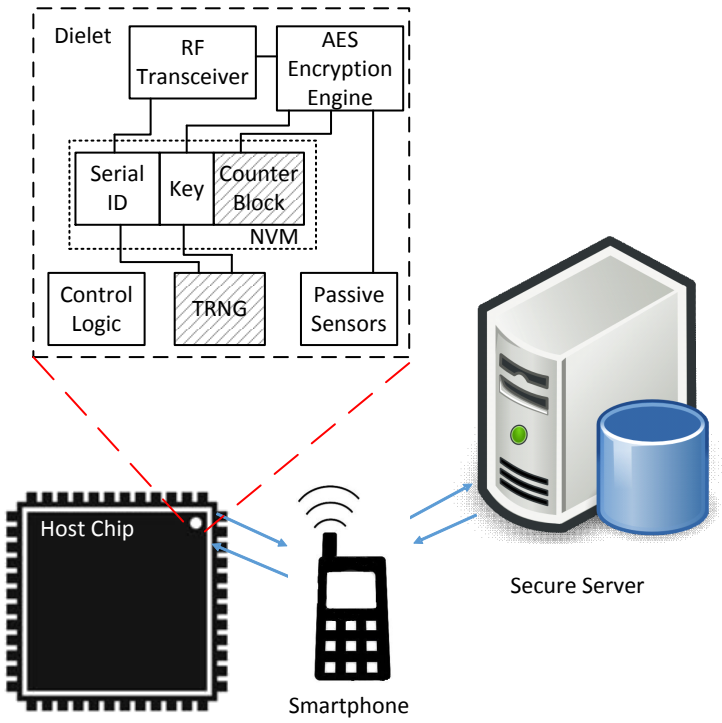

Figure 1: Authentication system of SHIELD. The shaded blocks are the modules that we propose to add.

of measuring extreme temperature, light exposure, vibration, UV radiation, etc. As a consequence of passive power supply, the sensors must be passive (e.g., light exposure can be recorded using photo sensitive material) [3].

The dielet has an encryption engine in order to encrypt the communication between the dielet and server. Some NVM (Non-volatile Memory) is embedded in the dielet to store a cryptographic key and a serial ID. We propose to add a hardware counter block (the top shaded block in Fig. 1) in NVM in order to enable AES (Advanced Encryption Standard) [4] counter mode encryption [5]. This as we will explain will make the authentication protocol more secure and will reduce the power consumption of the authentication protocol at the cost of one 8-bit counter in NVM as additional area overhead. In particular, we explain the *"try-and-check" attack which in case of DARPA's authentication protocol nullifies the effectiveness of SHIELD's main goal of being able to detect and trace adversarial activities with significant probability.* We show how our proposal prevents such an attack.

We also observe that the smartphone used in any authentication protocol *has to be trusted* to execute its role, because a compromised smartphone can deceive the user by displaying "authentication passed" on the screen without even getting in touch with the server. By assuming a trusted smartphone, our proposal reduces the number of communication rounds to only one.

Besides the authentication protocol we also need a protocol which initializes dielets with their own unique serial IDs and cryptographic keys. We propose to add a TRNG (True Random Number Generator) so that dielets can self-generate a serial ID and a secret key in parallel on the wafer during a trusted manufacturing process (the TRNG is depicted as the bottom shaded block in Fig.

1).

1.1 Contributions

We introduce the first formal framework of SHIELD protocols together with a taxonomy of attacks and corresponding adversarial security games in a concrete setting. Within this framework we introduce new authentication and initialization protocols for SHIELD with the following advantages:

1.1.1 Security Benefits

(a) Since DARPA's example authentication protocol uses plain AES which offers only *deterministic* symmetric key encryption, ciphertexts corresponding to the same plaintext in DARPA's protocol are linked over time. This makes their protocol particularly vulnerable to a simple *"try-and-check" attack which makes DARPA's protocol useless with respect to SHIELD's main aim*: The try-and-check attack nullifies an important role of the use of passive sensors in SHIELD in that adversaries are able to select which counterfeited/malicious chips have an assembled host package carrying a maliciously added dielet (taken from another legitimately produced host package) whose sensors did not detect any tampering (i.e., did not detect the removal and adding of the dielet from the legitimate host package to the maliciously assembled host package). So, adversaries are able to eliminate any trace or evidence of counterfeited/malicious chips that can be detected by the authentication protocol. Our proposal on the other hand prevents such attack, because it is based on AES in Counter Mode which is a probabilistic encryption scheme and therefore makes all ciphertexts look random (hence, unlinkable) to the adversary.
(b) The counter of AES in Counter Mode can be used as an extra sensor which teaches the authentication server how many times the dielet has been put into authentication mode when the dielet is offline with respect to the server. This can be used as an indicator of suspicious behavior.
(c) Due to a one-time initialization and two-phase activation construct in our initialization protocol, transits between trusted fabrication and trusted assembly facilities can be untrusted.

1.1.2 Performance Benefits

(a) We significantly reduce the dielet power consumption during authentication because of two crucial performance improvements:

- Our proposed authentication protocol transmits only 258 bits between the dielet and smartphone, instead of 448 bits in DARPA's example authentication protocol.
- Our proposed authentication protocol uses only one AES encryption per authentication request as opposed to DARPA's protocol which needs two AES encryptions per request.

(b) Our design needs one 8-bit counter in NVM: We notice that the write latency of many emerging non-volatile memories (STT-RAM, PCRAM and ReRAM) vary from 10's ns to 100's ns, while NAND FLASH, as the most widely used NVM technology nowadays, has a write latency of 200μs [6]. Nevertheless, the write latency of FLASH is still negligible compared to the network communication latency between the smartphone and remote server, which are in the tens or hundreds milliseconds [7]. Therefore the speed of any authentication protocol is dominated by the relatively large network latency between the smartphone and server. For this reason our protocol

achieves a 2× speed up compared to DARPA's protocol by reducing two full communication rounds between the smartphone and server in DARPA's protocol to only one round in our protocol.
(c) In our proposed initialization protocol dielets generate their own serial ID and key by using a true random number generator (TRNG). This allows dielets to efficiently generate their serial IDs and keys in parallel on the wafer during a trusted manufacturing process. The resulting performance overhead is insignificant. In our protocol dielets do need to be verified after transition from fabrication facility to assembly facility during a trusted assembly process to detect and prevent attacks on untrusted transits.

1.1.3 Area Utilization

(a) The additional area utilization for our authentication and initialization protocols is only 6% of the allowed area of the dielet ($0.01mm^2$ [2]) in 32nm technology (DARPA's example authentication protocol can be implemented in 2% of the allowed area). This is very small compared to the 55% of the allowed area needed for AES, leaving 39% of the allowed area for the passive sensors, RF transceiver, etc. using current state-of-the-art technology (for completeness, the main bottleneck is designing an RF transceiver which fits even a single mm^2).

1.2 Organization

Section 2 provides background on supply chain security, and discusses the existing supply chain vulnerabilities and current state-of-the-art countermeasures. Section 3 presents a formal framework of SHIELD protocols with a trust model, adversarial models, and formal security games. Section 4 explains DARPA's example authentication protocol and describes our proposal of a secure and efficient alternative together with a performance and security analysis. Section 5 introduces a new initialization protocol together with a performance and security analysis. Section 6 analyses the area overhead of our protocols by implementing the additional logic incurred by our protocols. Section 7 concludes the paper.

2 Supply Chain Security

2.1 IC Supply Chain Vulnerabilities

Supply Chain Security has recently gained significant interest in the hardware security community [8,9]. In [9], Guin *et al.* presented a detailed taxonomy of supply chain vulnerabilities, where supply chain vulnerabilities are classified into seven categories, as shown in Fig. 2. (1) *Cloned*: Cloning is a common threat for IC design. Adversaries want to reduce the cost of design by cloning the design of others and produce their own chips illegally. This may happen during the design phase (copying the design files illegally) or distribution phase (reverse engineering the chip to obtain the design of a chip). (2) *Tampered*: Tampering with a chip is more commonly known as adding a Hardware Trojan, a research area that gained lots of interest in recent years [10]. Each phase of the supply chain is vulnerable to an adversary inserting a Hardware Trojan in order to initiate stealthy and malicious or destructive behaviors. (3) *Overproduced*: Since the globalization of semiconductor business, fabrication and assembly have largely shifted to foundry/assembly facilities, these facilities are capable of producing more chips than those promised by a contract or agreement. Untrusted facilities will illegally sell overproduced chips for extra profit. (4) *Defective*: Defective chips fail to

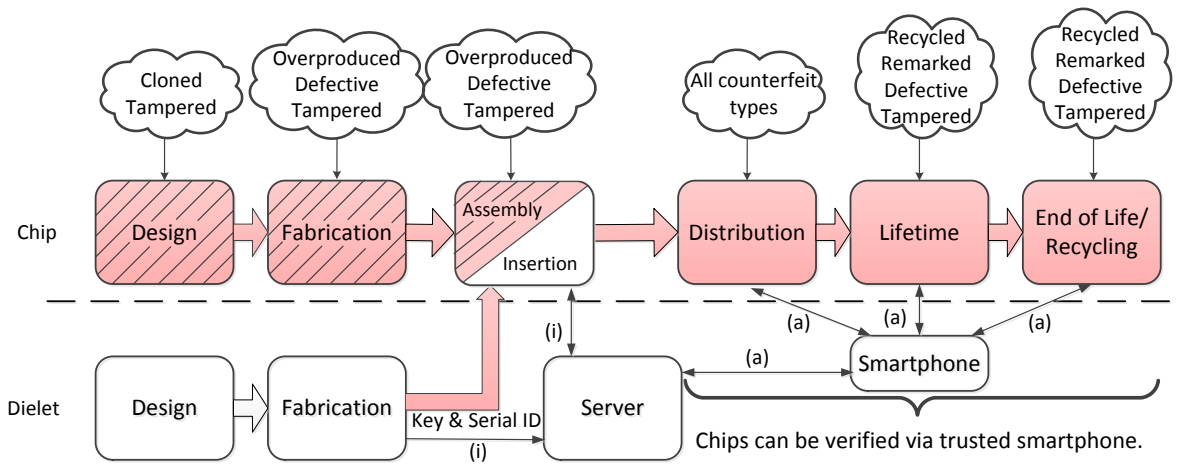

Figure 2: IC supply chain vulnerabilities and trust model for SHIELD. The white blocks indicate the trusted parties/facilities assumed by SHIELD. Specifically, dielet design, fabrication and insertion with initialization in IC host package are each trusted. The other parties in this figure are untrusted: we distinguish two types, those before insertion (dark boxes with diagonal lines) and those after insertion of dielets (dark boxes). Malicious behavior of parties after dielet insertion leading to counterfeited host chips is detected by the SHIELD authentication protocol. Detection of malicious behavior of parties before dielet insertion is out of SHIELD's scope and needs complementary techniques. The digital communication channels indicated by (a) between the dielets, smartphone, and the server are used in the authentication protocol. The digital communication channels indicated by (i) between the server and the dielet fabrication and insertion facilities are used by the initialization protocol. We assume that the server has established authenticated and secure digital communication channels with the dielet fabrication and insertion facilities and with the smartphones used in the authentication protocol.

correctly respond to at least one test vector. Yet, a defective chip may be sold as a functional chip on open markets by an irresponsible manufacturer who knows that the defect can only be excited in a corner case. (5) *Recycled*: After the lifetime of an electronic component, the component should be recycled properly. However, many recycled ICs are sold as new ICs after repackaging and remarking. (6) *Remarked*: In remarking, the adversaries remove the old marking on the chip and create a new coating for it with the aim to sell it with a higher specification. (7) *Forged Documentation*: Forged documentation includes a revision history of a component, certifications of compliance for some standards, etc.

2.2 Recent Detection/Avoidance Methods

Recently, researchers in academia and industry have proposed many counterfeit detection/ avoidance methods. Generally, these methods assign an ID to each chip and verify chips by checking these IDs. For example, silicon PUF (Physically Unclonable Function) technology [11, 12] can be used for IC identification and authentication since PUFs exploit the randomness of process variation during the fabrication phase to generate an unique fingerprint for each chip. It is impossible for an attacker to duplicate the PUF design with the exact same process variation to regenerate the same chip ID. This is used to target remarked, overproduced, and cloned chips in the supply

chain [9].

Another proposed countermeasure is Hardware Metering, which is a set of security protocols that enables design houses to achieve post-fabrication control of the fabricated IC. Usually hardware metering methods allow the designer to lock each chip through the fabrication, which means the chips are not functional, and these locked chips can be unlocked only by the designers [13]. This prevents overproducing and cloning. In similar style, Secure Split Testing allows designers to protect and meter their design after fabrication by introducing the designs into a testing phase to prevent defective chips from entering the open market [14].

Split Manufacturing is another method to prevent untrusted foundries from overproducing chips. The designers split the layout of a chip into front-end-of-line and back-end-of-line layers, and fabricate these two layers in two different untrusted foundries (who are assumed not to collaborate). Next, the design house just needs a low-end trusted manufacturing facility to assemble these two layers [15].

Finally, lightweight aging sensors can be embedded into chips to detect recycled chips [16]. Since these sensors are capable of recording the usage time of a chip, the verifier can easily figure out for how long this chip has been used.

3 The SHIELD Framework

3.1 Trust Model for SHIELD

The trust model for SHIELD is illustrated and explained in Fig. 2. The supply chain above the dashed line is the original supply chain for IC design, and the one below is added and corresponds to SHIELD, i.e., the production of trustworthy dielets which are inserted in the host packages of legitimately produced ICs. The design, fabrication, insertion and initialization of dielets need to be trusted (white blocks in Fig. 2). We assume a trusted transit between dielet design and fabrication, while all other transits between functional facilities can potentially be untrusted including the transit between dielet fabrication and insertion. Smartphones used as an intermediary for verification must be trusted, since a compromised smartphone can deceive the human user of the smartphone by displaying “authentication passed" at the very end of the authentication protocol even if the dielet did not pass or would not have passed authentication.

Fig. 2 completes initialization of dielets after their insertion into a host package, the reason being that dielets need to be *activated* after their insertion in order to avoid triggering of the passive sensors during dielet fabrication and insertion.

Dielets allow the identity and authenticity of chips to be verified at any stages in the supply chain after the insertion process. Notice that even if the manufacturing facility (i.e., chip fabrication and assembly) cannot be trusted, overproduction can now be prevented by controlling and recording the number of used dielets. Therefore, SHIELD alone without assuming a trustworthy manufacturing facility for ICs has a high coverage of vulnerabilities after the dielets are inserted and activated in assembly facilities. To protect the entire supply chain against supply chain security vulnerabilities, SHIELD can be combined with complementary security and testing techniques, such as post-silicon Hardware Trojan detection [17,18] and VLSI (Very Large Scale Integrated circuit) testing schemes [19] before the insertion of dielets. Note that how to cover the security issues in chip design, fabrication, and assembly facilities is out of SHIELD’s scope.

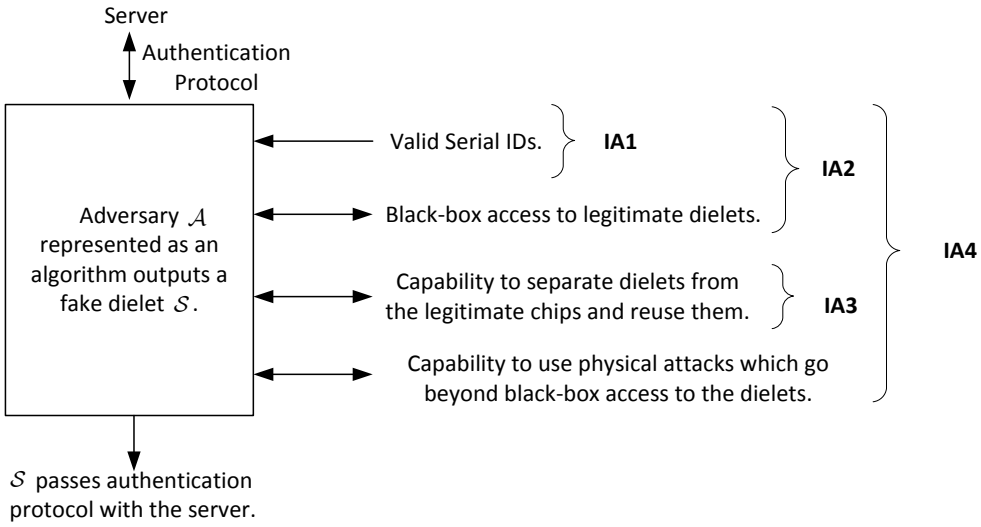

Figure 3: The adversarial models of Impersonation Attacks (IA).

3.2 Adversarial Models

We classify all legitimate dielets into (1) Non-validated dielets, which have not yet been validated by the authentication server at the end of the initialization protocol (and as a consequence will not be recognized or authenticated by the server), (2) Validated dielets, which have finished the initialization process completely and will be authenticated by the server in future authentication sessions.

We have two types of attacks: (1) Denial-of-Service attacks, which corrupt legitimate dielets, and (2) Impersonation attacks, which circumvent the authentication protocol by a fake dielet or chip. Notice that these attacks can be applied to both validated and non-validated dielets.

1. **DoS; Denial-of-Service Attacks:** The purpose of this attack is to make a legitimate SHIELD dielet not function correctly in that its host chip cannot pass authentication. An example could be triggering the passive sensors intentionally by physical attacks. Notice that *single dielet denial-of-service*, i.e. a denial-of-service attack which is performed to dielets one by one sequentially, is allowed by SHIELD; this is because SHIELD adds passive sensors into dielets and discards the chip if the sensors indicate the occurrence of physical attacks, hence, if in physical proximity, legitimate dielets can be corrupted one by one. However, we do need to prevent *batch mode denial-of-service* which corrupts a batch of legitimate dielets simultaneously; the dielet interface should only accept communication from the initialization or authentication protocols and each protocol interaction should be personalized with respect to the dielet.

In contrast to denial-of-service, impersonation attacks try to inject counterfeit or even malicious chips into the supply chain with dielets that verify/authenticate properly. Fig. 3 shows several impersonation attacker models:

2. **IA1; The attacker receives valid serial IDs:** Each legitimate dielet is associated to a

unique valid serial ID. The attacker has a list of valid serial IDs and wants to produce a fake dielet associated to one of the valid serial IDs, in order to pass one authentication.

3. **IA2; The attacker has oracle access to legitimate dielets:** Attackers in this category also have black-box access to legitimate dielets. Collected dielet responses are used to produce a fake dielet that can pass one authentication.

4. **IA3; The attacker has the capability to separate a dielet from its legitimate host chip and reuse the dielet in another chip of his choice such that with positive probability $1-\rho > 0$ none of the dielet's sensors sensed/detected the separation or reuse, as we denote ρ as the probability of being detected:** In this model the attacker is able to perform invasive physical attacks to open host packages and separate dielets from host chips. Once separated the dielets can be glued to host packages of counterfeit chips.

5. **IA4; The attacker extracts secret keys from dielets by performing physical attacks:** This is the strongest adversarial model where the attacker is able to go beyond black-box access to dielets to extract the internal secret keys, e.g. side channel analysis [20, 21], differential fault analysis [22, 23] or probing attacks [24, 25].

3.3 SHIELD Protocols

The three parties of a SHIELD system are defined as follows:

Dielet: The state of a dielet $\mathcal{D}$ is represented by a tuple (ID, Key, xxx, SS), where $SS \in \{''OK'', \perp\}$ represents sensor status bits with $''OK''$ the initial state indicating an all-is-fine signal and $\perp$ indicating a physical attack has been detected, and where "xxx" represents other possible states corresponding to a specific dielet construction. We assume that the secret key Key of each dielet is uniform random.

Smartphone: A smartphone $\mathcal{P}$ works as an intermediary between a dielet $\mathcal{D}$ and server. $\mathcal{P}$ shares an authenticated and secure communication channel with the server. Note that the smartphone itself is trusted, i.e. its execution follows protocol design and cannot be altered by adversaries.

Server: The server or verifier $\mathcal{V}$ shares an authenticated and secure communication channel with each smartphone $\mathcal{P}$ and has a database to store authentication information entries (ID, Key, xxx). $\mathcal{L}(\mathcal{V})$ denotes server $\mathcal{V}$'s current list of serial IDs of (validated) dielets which have completed their initialization process (this list is dynamically updated), and $\mathcal{K}(\mathcal{V})$ represents the list of secret keys associated to each serial ID in $\mathcal{L}(\mathcal{V})$. Note that the server is assumed to be trusted in our trust model, hence, at most the content of $\mathcal{L}(\mathcal{V})$ should possibly be leaked to any adversary through (protocol) interaction with $\mathcal{V}$; serial IDs are used to start protocol interactions and are therefore not encrypted and can be viewed by adversaries while other information in $\mathcal{V}$'s database should only be used internally to $\mathcal{V}$ in order to register and verify authenticity of dielets. Assuming these design principles, we do not give adversary $\mathcal{A}$ oracle access to $\mathcal{V}$ explicitly, instead in our adversarial models and security games we assume $\mathcal{L}(\mathcal{V})$ to be public to $\mathcal{A}$ and we assume $\mathcal{A}$ to be able to play the role of dielet and smartphone in authentication protocol instances with the server.

A complete SHIELD system consists of two protocols:

- The *initialization protocol* $Init$: $(\mathcal{D}, \mathcal{V}_{new}) \leftarrow Init(\mathcal{V})$ generates $\mathcal{D}$. $\mathcal{V}$ is updated, in particular $\mathcal{L}(\mathcal{V})$ and $\mathcal{K}(\mathcal{V})$ are independently updated. Each element in $\mathcal{K}(\mathcal{V})$ should be uniformly

distributed and independently generated, and each element in $\mathcal{L}(\mathcal{V})$ should be unique. At the end of this process, dielet $\mathcal{D}$ is embedded into an IC host package with all passive sensors activated.

- The *authentication protocol Auth:* $(\mathcal{V}_{new}, \mathcal{D}_{new}, ok) \leftarrow Auth(\mathcal{D}, \mathcal{P}, \mathcal{V})$, where $ok \in \{0, 1\}$. The output ok of the authentication protocol equals 1 if and only if $\mathcal{D}$ passes the authentication protocol. The state of dielet $\mathcal{D}$ (represented by xxx above) can be modified during the authentication protocol leading to $\mathcal{D}_{new}$.

Correctness: With respect to *correctness* (i) if $SS = \perp$ or $ID \notin \mathcal{L}(\mathcal{V})$ (i.e. the ID is invalid) for dielet $\mathcal{D}$, then $ok = 0$, and (ii) if $(\mathcal{D}, \mathcal{V}_{new}) \leftarrow Init(\mathcal{V})$, then $(\mathcal{V}_{new}, \mathcal{D}_{new}, 1) \leftarrow Auth(\mathcal{D}, \mathcal{P}, \mathcal{V})$.

3.4 Security Games

We define several security games corresponding to impersonation attacks under IA1, IA2, and IA3. In our first game, given m legitimate dielets, the adversarial algorithm $\mathcal{A}$ generates a simulation algorithm $\mathcal{S}$ which represents a fake dielet.

Game IA1$(t, m, \mathcal{A})$
  Initialize $\mathcal{V}_0$
  for $i \leftarrow (1, m)$ **do**
    $(\mathcal{V}_i, \mathcal{D}_i) \leftarrow Init(\mathcal{V}_{i-1})$
  end for
  $\mathbb{D} = (\mathcal{D}_1, ..., \mathcal{D}_m)$
  Generate fake dielet $\mathcal{S} \leftarrow \mathcal{A}^{Auth(.,.,\mathcal{V}_m)}(\mathcal{L}(\mathcal{V}_m))$ within t computation steps
  $(\mathcal{V}_{new}, \mathcal{S}_{new}, ok) \leftarrow Auth(\mathcal{S}, \mathcal{P}, \mathcal{V}_m)$
  Return ok ▷ $ok = 1$ means that the adversary wins
Game end

In IA1 $\mathcal{A}$ is given access to the authentication protocol where he can play the role of dielet and smartphone while interacting with server $\mathcal{V}_m$.

Definition 1. *We define $(Init, Auth)$ to be* IA1 *$\epsilon-$secure if for all $(t, m, \mathcal{A})$ the probability of wining game* IA1

$$Pr[\text{IA1}(t, m, \mathcal{A}) = 1] \leq \epsilon(t, m),$$

where ϵ is a function of t and m.

In our second game the adversarial algorithm $\mathcal{A}$ has oracle access to $\mathbb{D} = (\mathcal{D}_1, ..., \mathcal{D}_m)$. This represents black-box access to legitimate dielets which $\mathcal{A}$ uses for "reverse-engineering" (part of) the internal states of $\mathbb{D}$.

Game IA2$(t, m, \mathcal{A})$
  Initialize $\mathcal{V}_0$
  for $i \leftarrow (1, m)$ **do**
    $(\mathcal{V}_i, \mathcal{D}_i) \leftarrow Init(\mathcal{V}_{i-1})$
  end for
  $\mathbb{D} = (\mathcal{D}_1, ..., \mathcal{D}_m)$
  Generate fake dielet $\mathcal{S} \leftarrow \mathcal{A}^{Auth(.,.,\mathcal{V}_m),\mathbb{D}}(\mathcal{L}(\mathcal{V}_m))$ within t computation steps
  $(\mathcal{V}_{new}, \mathcal{S}_{new}, ok) \leftarrow Auth(\mathcal{S}, \mathcal{P}, \mathcal{V}_m)$

Return ok ▷ $ok = 1$ means that the adversary wins
Game end

Definition 2. *We define* $(Init, Auth)$ *to be* IA2 $\epsilon-$*secure if for all* $(t, m, \mathcal{A})$ *the probability of wining game* IA2

$$Pr[\text{IA2}(t, m, \mathcal{A}) = 1] \leq \epsilon(t, m).$$

For IA-3, we need to define a separation process, because we allow the adversary to reuse separated legitimate dielets in fake dielets as subroutines:

- The *separation process Sep:* $\mathcal{D}' \leftarrow Sep(\mathcal{D})$ separates the dielet $\mathcal{D}$ from the host chip. After this process, $Prob[SS = \perp] = \rho$, which means the physical separation is detected with probability ρ. No sensor technology is full proof and for this reason $1 - \rho > 0$ and cannot be negligibly small. For example, manual material removal is mentioned to be much more effective than might be thought in [26]: "A dexterous attacker is able to accomplish extremely delicate work without tripping a sensor." In addition, optical sensors can sometimes be bypassed by launching attacks in a dark room or dropping some black ink over the optical sensors [27].

Game IA3$(t, m, \mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1))$
  Initialize $\mathcal{V}_0$
  for $i \leftarrow (1, m)$ **do**
    $(\mathcal{V}_i, \mathcal{D}_i) \leftarrow Init(\mathcal{V}_{i-1})$
  end for
  $\mathbb{D} = (\mathcal{D}_1, ..., \mathcal{D}_m)$ and $\mathcal{L}(\mathcal{V}_m)$ for $\mathbb{D}$ is generated as well
  Information $X \leftarrow \mathcal{A}_0^{Auth(.,.,\mathcal{V}_m),\mathbb{D}}(\mathcal{L}(\mathcal{V}_m))$ is collected within $t_0 \leq t$ computation steps
  for $i \leftarrow (1, m)$ **do**
    $\mathcal{D}'_i \leftarrow Sep(\mathcal{D}_i)$
  end for
  Generate $\mathcal{S}^{\mathbb{D}'} \leftarrow \mathcal{A}_1^{Auth(.,.,\mathcal{V}_m),\mathbb{D}'}(\mathcal{L}(\mathcal{V}_m), X)$ within $t_1 = t - t_0$ computation steps
  $(\mathcal{V}_{new}, \mathcal{S}^{\mathbb{D}'}_{new}, ok) \leftarrow Auth(\mathcal{S}^{\mathbb{D}'}, \mathcal{P}, \mathcal{V}_m)$
  Return ok ▷ $ok = 1$ means that the adversary wins
Game end

Note that $\mathcal{S}^{\mathbb{D}'}$ represents a fake dielet with the devices $\mathbb{D}'$ embedded into it. The fake dielet $\mathcal{S}^{\mathbb{D}'}$ uses the embedded $\mathbb{D}'$ as an oracle and interacts with it as a black box. Game IA3 represents an extremely capable and in this sense (as of now completely) unrealistic attacker: Notice that we do not consider physical IA4 attacks. So, if the attacker wishes to use $\mathcal{S}$ circuitry, then devices $\mathbb{D}'$ must interact through their transceivers with $\mathcal{S}$, implying that $\mathcal{S}$ must have a transceiver and the devices must be embedded such that no other wireless signal to and from a smartphone interferes with the combined $\mathcal{S}$ with $\mathbb{D}'$ functionality, yet devices $\mathbb{D}'$ must be powered-up through their transceivers. For completeness, if at the end of the separation process devices $\mathbb{D}'$ have been invasively altered in a IA4 attack in order to add $\mathcal{S}$ to replace part of the dielet's internal functionality, then ρ will become very close to 1. In fact, if an attacker is able to perform such invasive probing, then he would be able to extract a secret key from a dielet and use this to produce $\mathcal{S}$.

In the next section we show how a realistic and therefore much weaker IA3 attacker, who simply uses a separated dielet as a fake dielet (without any alterations or additional $\mathcal{S}$ circuitry) and glues it to the host package of a counterfeit chip, defeats DARPA's example authentication protocol. Here, we may assume that gluing a separated dielet to another package can be done in

such a careful way that no more sensors in the separated dielet will trip; this assumption gives the adversary an advantage and is therefore a worst-case assumption for the defender, see Sec. 4.1 for a detailed discussion. With respect to the next security definition we show that our solution is able to deal with this weak-IA3 attack and we explain a small modification which can cope with even the most unrealistically capable IA3 attacker.

Definition 3. *We define* $(Init, Auth)$ *to be* IA3 $\epsilon-$*secure if for all* $(t, m, \mathcal{A})$ *the probability of wining game* IA3 *where* $\mathcal{A}$ *has no oracle access to* $Auth(.,.,\mathcal{V}_m)$ *is*

$$Pr[\text{IA3}(t, m, \mathcal{A}) = 1] \leq (1 - \rho) + \epsilon(t, m).$$

Since the passive sensors can only detect physical separation with probability ρ, game IA3 can be won with probability $\geq 1 - \rho$. If $\mathcal{A}$ engages in an authentication protocol, then he can verify whether a separated dielet passes authentication, i.e., whether its sensors detected the physical separation. So, if $\mathcal{A}$ has oracle access to $Auth(.,.,\mathcal{V}_m)$, then IA3 can be won with probability 1. In each of the two cases (with or without engaging in an authentication protocol), the server will notice a failed authentication with probability $\approx \rho$. This means a trace of adversarial behavior will be detected by the server. In the next section we show that DARPA's example protocol does not resist a weak-IA3 attacker implying the existence of a successful attacker who does not leave any trace for the authentication server to detect.

Security Analysis IA4: Within the limited amount of budget of each dielet (DARPA requires the area of a dielet to be 0.01 mm^2 [2]), it is almost impossible to have countermeasures against IA4, even area efficient countermeasures such as [28, 29] are prohibitive. We can only try our best to demotivate economically-motivated adversaries by maximizing their cost to produce a counterfeit chip.

We define this cost σ as $\sigma = \frac{\beta}{\delta}$, where β is the cost of producing a fake dielet and δ is the number of successful authentication attempts over the life time of a fake dielet. Overall, σ measures the cost of a single successful authentication. If we can minimize δ and maximize β, then an attacker will have minimal economical gain by performing IA4. Adding countermeasures against IA4 attacks, as a conventional solution, increases β, because the cost of attack increases. But given the extremely low budget, we propose to set an upper bound on the number of times a dielet can successfully engage in an authentication protocol to a reasonable value. This limits the usage of each dielet which upper bounds δ.

4 Authentication Protocol

4.1 DARPA's Protocol

In the call for proposals for SHIELD [2], DARPA suggests the authentication protocol as depicted in Fig. 4. In order to verify the authenticity of the host chip, a smartphone with an inductive or RF probe plugged into it is used to first power up the dielet and upload the serial ID of the dielet to the server. Next, the server looks up the entry that corresponds to the serial ID in its database. If it is an existing serial ID in the database, the server generates a random challenge/nonce C and sends C to the dielet through the smartphone. The dielet separately encrypts the challenge C and the sensor status bits SS by using AES with the on-board cryptographic key K, and sends the resulting ciphertexts X and Y back to the server. The server decrypts the ciphertext X using the

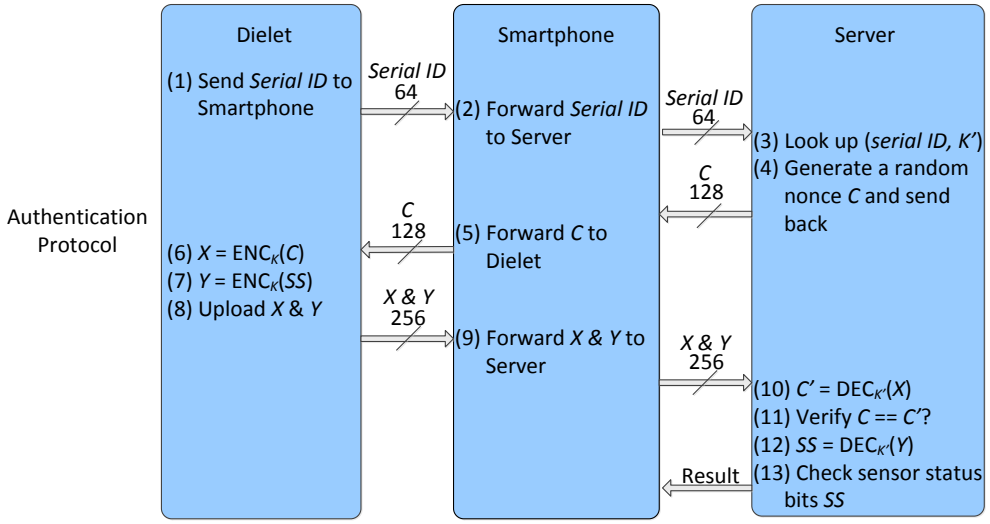

Figure 4: DARPA's Authentication Protocol. SS is sensor status. There are two full communication rounds between the smartphone and the remote server. It encrypts the challenge and sensor status bits separately. The total transmitted bits between the dielet and the smartphone are 448 bits.

key K' corresponding to the serial ID stored in the server's database, and compares the result C' with the original challenge C. If C' matches C and the sensor status bits do not record any attacks, then the server concludes that this chip is legitimately produced. In the last step the server replies the authentication result to the smartphone (to avoid man-in-the-middle attacks the smartphone and server are assumed to have their own authentic communication channel).

Without proofs, we state that DARPA's protocol is IA1 and IA2 $\epsilon-$secure for practically small enough functions ϵ. However, the protocol is not weak-IA3 $\epsilon-$secure: The following algorithm describes a "try-and-check" attack. Notice that an IA3 attack allows reuse of dielets separated from legitimate chips.

Algorithm Try-and-Check Attack($\mathbb{D}$)
  for All $\mathcal{D}_i$ in $\mathbb{D}$ **do**
    $(X_i, Y_i) \leftarrow \mathcal{D}_i(C_i)$ ▷ Simulate $Auth(\mathcal{D}_i, \mathcal{P}, \mathcal{V})$ and store $(C_i, (X_i, Y_i))$
    $\mathcal{D}'_i \leftarrow Sep(\mathcal{D}_i)$
    $(X'_i, Y'_i) \leftarrow \mathcal{D}'_i(C_i)$
    if $X_i \neq X'_i$ or $Y_i \neq Y'_i$ **then**
      Remove $\mathcal{D}'_i$ from $\mathbb{D}'$
    end if
  end for
  Return $\mathbb{D}'$
Algorithm end

This "try-and-check" attack *nullifies the effectiveness of DARPA's authentication protocol in that the protocol becomes useless in one very important aspect*: Suppose that a dielet embedded in a legitimate host package is separated from the host package and is inserted into, or glued or added to a maliciously created host package of a counterfeit chip or a chip with Hardware Trojan. The dielet is designed such that with significant probability $\rho > 0$ its passive sensors detect the

physical attack that separates it from its host. The adversary, however, may *try* in the hope that the separation from a legitimate chip's host package and insertion into the host package of a counterfeit or malicious chip is not detected by the passive sensors (with probability $1-\rho$). In DARPA's protocol the adversary can apply the same challenge before and after the physical attack to *check* whether the replied ciphertext has changed or not. If the ciphertext remains unchanged, then this indicates that the physical attack was not detected by the passive sensors which would otherwise have led to a change in the sensor status bits and the ciphertext. So, the attacker is able to find out which of the counterfeit chips will be verified correctly by DARPA's authentication protocol. This means that the attacker is able to figure out which counterfeit or malicious chips can be put into the supply chain without the authentication server being able to detect anything suspicious. By repeating this process on all the legitimate dielets $\mathbb{D}$, the "try-and-check" attacker is able to get a set of successfully-separated dielets $\mathbb{D}'$ which can pass the authentication without being detected.

Hence, a "Try-and-Check" attacker $\mathcal{A}_{tc}$ can successfully produce a fake host chip to win this game: For $t, m \geq 1/(1-\rho)$,

$$Pr[\text{IA3}(t, m, \mathcal{A}_{tc}) = 1] \approx 1.$$

Even though the passive sensors make it hard (with only a success probability $1-\rho$) to maliciously extract a small part of a legitimate host package that contains the host package's dielet and add this part to a maliciously assembled host package for a counterfeit chip or a chip with a Hardware Trojan *without the dielet's passive sensors detecting the malicious activities*, the attack nullifies the effectiveness of the main role of passive sensors in SHIELD: at least some trace or evidence (preferably a fraction of ρ counterfeited/malicious chips that are being put back into the supply chain) should survive.

Also notice that the above attack offers an adversary the opportunity to use its counterfeit chips in a cost-effective way, i.e., they will only be inserted into the supply chain if the added dielet did not detect suspicious behavior. In other words no counterfeit chips are lost which reduces the cost of attack.

As a final remark: A strong (unrealistic) IA3 attacker can defeat DARPA's protocol in a second way. Since C and SS are encrypted in different ciphertexts, an attacker may first retrieve a ciphertext Y corresponding to a "clean" sensor status SS. After separation of a dielet the attacker embeds it in $\mathcal{S}$ circuitry which intercepts and replaces any newly outputted ciphertext Y' with the retrieved Y. The authentication server will decrypt Y and conclude no physical attack was detected by the dielet sensors.

4.2 Our Solution

As explained in the introduction, we propose to base the authentication protocol on AES counter mode: Counter (CTR) mode is a mode of operation for block ciphers introduced by Diffie and Hellman in 1979 [5, 30]: In order to produce a ciphertext, the most recently used counter value is incremented and encrypted, and the result is exclusive-ORed with the plaintext. We notice that the NSA (National Security Agency) Suite B Cryptography approved AES in CTR mode [31] and that AES counter mode is also recommended by NIST (National Institute of Standards and Technology) [32]. E.g., counter mode encryption is used in the IPsec Internet Draft [33] and ATM Security Specification [34].

Before we explain our authentication protocol, we notice that counter mode encryption in dielets has a potential vulnerability in that it can be exploited in a denial-of-service (DoS) attack:

An attacker can power up a batch of dielets, causing each dielet to irreversibly increase its counter. This may force counters to run out of range, i.e., the server will not be able to synchronize its own copies of the counter values and will therefore reject future legitimate attempts of dielets to authenticate themselves. In order to prevent significant loss (of many dielets) due to such an efficient *batch-mode* DoS attack, we add a read-out mode before the authentication mode. Only after the dielet confirms (by verifying a challenge) that the smartphone attempts to transmit messages with this specific dielet, the dielet will enter an authentication mode during which its counter is incremented.

We notice that by adding a read-out mode before the authentication mode, only batch-mode DoS attacks are prevented. I.e., a single-dielet DoS attack is still possible, which is consistent with SHIELD's philosophy, see Sec. 3.2 .

Although the counter value itself can be used as a built-in challenge for this authentication protocol, a random nonce from the trusted verifier side is still required to guarantee the freshness of the authentication messages, in order to prevent replay attacks.

4.3 Read-out Mode

Initially, during read-out mode, the dielet waits to be powered up. As soon as it powers up, the dielet checks whether the counter value CB is larger than 1 and less than a maximum value MAX. If the check fails, then either the dielet has not yet been fully initialized (CB equals 0 or 1) and should enter self-generation and initialization mode (see section 5), or the counter has reached a prefixed maximum number MAX of times the dielet is allowed to enter authentication mode (and during which its sensor status SS is read-out, encrypted and communicated to the server). Checking CB being less than MAX limits the number of encryptions on the dielet and prevents the counter to roll-over and repeat itself.

If the check passes, then the dielet will proceed to the next step (15). See Fig. 5 (c), the dielet transmits its serial ID to the smartphone via the read-out unit. The smartphone truncates the serial ID to L bits and sends the truncated serial ID together with a uniformly distributed M-bit random nonce C back to the dielet. The dielet verifies the reply from the smartphone which, if correct (i.e., it corresponds to the dielet's serial ID), puts itself into authentication mode. Because the smartphone has to send the truncated L-bit serial ID back to confirm the communication request, DoS attacks against a batch of dielets *simultaneously* are prevented (since each dielet verifies its own unique L-bit truncated serial ID).

4.4 Authentication Mode

Fig. 5 (d) shows the sequence of operations in authentication mode. After the dielet enters authentication mode, the on-board cryptographic key K and the hardware counter block CB are retrieved from NVM to compute $X = AES_K(C||CB)$ [1]. At the same time, the counter block CB is incremented. In step (19) the dielet takes the first N bits of X and exclusive-ORs these with the S-bit status bits (SS) from the dielet's passive sensors padded with zeroes up to N bits ($N \ll 128$). The padded zeros are used for authentication as explained below. The result after the exclusive-OR is an N-bit "verification message" V which is transmitted to the smartphone. After the above steps, the dielet returns to read-out mode, and waits for a next authentication request in step (14).

[1] Here "||" means bit concatenation. This is the construction of a counter value suggested by NIST in [32]

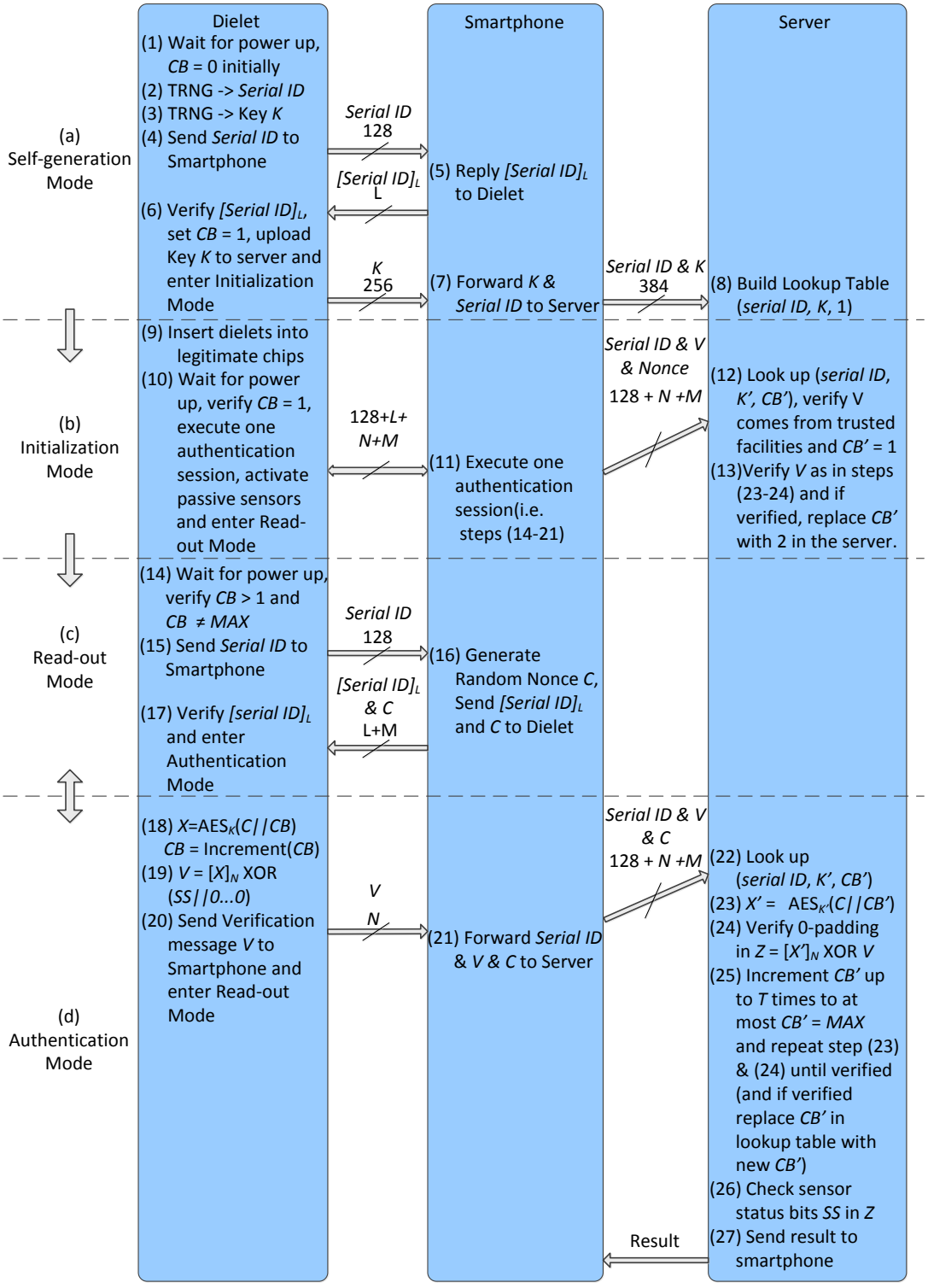

Figure 5: Proposed Authentication ($Auth$) and Initialization ($Init$) Protocols for SHIELD. (a) and (b) are the self-generation and initialization modes of the initialization protocol, while (c) and (d) are the read-out and authentication modes of the authentication protocol. Smartphone and server are assumed to be trusted.

The smartphone forwards V to the authentication server together with the serial ID and the random nonce C used for this dielet in this authentication session. The authentication server looks up the key K' and counter block CB' associated to the serial ID and computes $X' = AES_{K'}(C||CB')$. The first N bits of X' are exclusive-ORed with V. Without malicious behavior, the result Z should be equal to the string SS padded with zeroes. If Z indeed shows all of the padded zero bits, then the server concludes that the ciphertext was produced by the dielet with the serial ID. In addition, the server concludes that the first S non-zero bits of Z are the sensor status bits of the dielet's passive sensors.

The above protocol assumes that the counter block stored at the authentication server CB' and the dielet's counter block CB are synchronized. This may not be the case in practice due to potential network issues, e.g. disconnection with the server, and for this reason the authentication server repeatedly increases CB' and repeats steps (23) and (24) until either authentication passes (and CB' in the look up table is replaced by the increased CB') or the number of increments goes beyond a pre-fixed threshold T in which case authentication fails (and the server resets the counter block to its originally stored value). After verifying the padded zeros and sensor status bits, the server sends the authentication result to the smartphone. Fig. 5 (c) and (d) show the complete authentication protocol.

4.5 Security Analysis

Before proving the security of our authentication protocol, we define the security of AES encryption. Since SHIELD is a concrete system, we concretely set security parameters rather than using asymptotic notation.

We first modify DEFINITION 3.23 in [35] from an asymptotic setting to a concrete setting:

Definition 4. *Let* $AES_K(.) : (x, K) \in \{0,1\}^{128} \times \{0,1\}^{256} \rightarrow y = AES_K(x) \in \{0,1\}^{128}$ *be an efficient, keyed function. We say that* $AES_K(.)$ *cannot be distinguished from a random permutation with advantage* $> \epsilon_{AES}(t, q)$ *if for all distinguishers* $\mathcal{A}$ *which execute within* t *computation steps and make at most* q *oracle queries,*

$$|Pr[\mathcal{A}^{AES_K(.)} = 1] - Pr[\mathcal{A}^{g(.)} = 1]| \leq \epsilon_{AES}(t, q),$$

where $K \leftarrow \{0,1\}^{256}$ *is chosen uniformly at random and* g *is chosen uniformly at random from the set of* permutations *mapping 128-bit strings to 128-bit strings.*

Definition 5. *Let* $[AES_K(.)]_N : \{0,1\}^{128} \times \{0,1\}^{256} \rightarrow \{0,1\}^N$ *be the function that outputs the first* N *bits of* $AES_K(.)$*, where integer* $0 < N \leq 128$*. We say* $[AES_K(.)]_N$ *cannot be distinguished from a random function with advantage* $> \epsilon_{tAES}(t, q, N)$ *if for all distinguishers* $\mathcal{A}$ *which execute within* t *computation steps and make at most* q *oracle queries,*

$$|Pr[\mathcal{A}^{[AES_K(.)]_N} = 1] - Pr[\mathcal{A}^{f(.)} = 1]| \leq \epsilon_{tAES}(t, q, N),$$

where $K \leftarrow \{0,1\}^{256}$ *is chosen uniformly at random and* f *is chosen uniformly at random from the set of random* functions *mapping 128-bit strings to* N*-bit strings.*

Combining Theorem 1 and Theorem 2 in [36], we know that the probability of distinguishing a truncated random permutation from a random function can be stated as follows:

Lemma 1. *[36] Let $0 < m \leq n$ be integers. For all distinguishers $\mathcal{A}$ with at most q queries to an oracle, which is either a truncated random permutation $g(.) : \{0,1\}^n \to \{0,1\}^m$ or a random function $f(.) : \{0,1\}^n \to \{0,1\}^m$, the advantage*

$$Adv_{tRP}(m,n,q) = |Pr[\mathcal{A}^{g(.)} = 1] - Pr[\mathcal{A}^{(f(.)} = 1]|$$

of distinguishing these oracles can be upper bounded as a function of $p = q \cdot 2^{-(n-m/2)}$:

$$Adv_{tRP}(m,n,q) \leq \begin{cases} 2\sqrt[3]{2}p^{\frac{2}{3}} + \frac{2\sqrt{2}}{\sqrt{3}}p^{\frac{3}{2}} + p^2, & \text{if } \frac{2n}{3} \leq m \leq n \\ 3p^{\frac{2}{3}} + 2p + 5p^2 + \frac{1}{2}(2p)^{\frac{n}{m}}, & \text{if } 4 + \log_2 n \leq m \\ & \quad < \frac{2n}{3}. \end{cases}$$

By Definitions 4 and 5, and Lemma 1, we have

Lemma 2. *For $0 < N \leq 128$,*

$$\epsilon_{tAES}(t,q,N) \leq \epsilon_{AES}(t,q) + Adv_{tRP}(N,128,q).$$

For the security analysis of IA3, we adapt the definition of CPA (Chosen-Plaintext Attack) indistinguishability, see e.g. [35], towards the concrete setting for AES in CTR mode:

1: **Algorithm** $[\text{AES-CTR}_K]_N(C,x)$
2: Increment counter CB
3: Return $[AES_K(C||CB)]_N XOR\ x$
4: **Algorithm end**

CPA indistinguishability experiment $CPA(N,t,q,\mathcal{A})$

1. A key K is chosen uniformly at random from $\{0,1\}^{256}$.
2. The adversary $\mathcal{A}$ has $q_0 \leq q$ oracle accesses to $[\text{AES-CTR}_K]_N$ and outputs a pair of messages $x_0, x_1 \in \{0,1\}^N$ together with a pair of challenges C_0, C_1 of the same length within $t_0 \leq t$ computation steps.
3. A random bit $b \leftarrow \{0,1\}$ is chosen and then a ciphertext $v \leftarrow [\text{AES-CTR}_K]_N(C_b, x_b))$ is computed and given to $\mathcal{A}$.
4. $\mathcal{A}$ continues to have $q_1 = q - q_0$ oracle accesses to $[\text{AES-CTR}_K]_N$ and outputs a bit b' within $t_1 = t - t_0$ computation steps.
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Definition 6. *We say the CPA indistinguishability experiment can be won with advantage at most $\epsilon_{cAES}(t,q,N)$ if for all adversaries $\mathcal{A}$*

$$Pr[1 \leftarrow CPA(N,t,q,\mathcal{A})] \leq \frac{1}{2} + \epsilon_{cAES}(t,q,N).$$

From case 1 of the proof of Theorem 3.29 in [35] we infer the next lemma:

Lemma 3. *[35] If the counter value CB is never repeated, then the CPA indistinguishability experiment can be won with advantage at most $\epsilon_{cAES}(t,q,N) = \epsilon_{tAES}(t,q,N)$.*

By using the above definitions and bounds we prove Theorem 1 below in the remainder of this subsection and we conclude this subsection with a discussion on IA4.

Theorem 1. *The proposed authentication protocol is* IA1 ϵ_1-*secure,* IA2 ϵ_2-*secure and weak-*IA3 ϵ_3-*secure, where*

$$\begin{aligned}\epsilon_1 &= \epsilon_{tAES}(t,0,N)+\frac{T}{2^N},\\ \epsilon_2 &= \epsilon_{tAES}(t,MAX,N)+\frac{T}{2^N}+\frac{T}{2^M},\ \textit{and}\\ \epsilon_3 &= 2\epsilon_{tAES}(t,MAX,N).\end{aligned}$$

4.5.1 IA1 Security

An IA1 attacker $\mathcal{A}$ wants to generate a fake dielet $\mathcal{S}$ just based on the knowledge of valid serial IDs $\mathcal{L}(\mathcal{V}_m)$ with the property that $\mathcal{S}$ passes the verification protocol $(\mathcal{V}_{new},\mathcal{S}_{new},1)\leftarrow Auth(\mathcal{S},\mathcal{P},\mathcal{V}_m)$. We first note that by the definition of initialization protocol $Init(.)$, the generation of $\mathcal{L}(\mathcal{V}_m)$ is statically independent of the generation of the keys $\mathcal{K}(\mathcal{V}_m)$. Also, $\mathcal{L}(\mathcal{V}_m)$ does not give any information about plaintext-ciphertext pairs. So, knowledge of $\mathcal{L}(\mathcal{V}_m)$ only helps the adversary in selecting a valid serial ID for $\mathcal{S}$, and we can change the security game by replacing $\mathcal{A}^{Auth(.,.,\mathcal{V}_m)}(\mathcal{L}(\mathcal{V}_m))$ by just $\mathcal{A}^{Auth(.,.,\mathcal{V}_m)}(ID)$, where ID is a valid serial ID in $\mathcal{L}(\mathcal{V}_m)$.

See Fig. 5, in order to pass verification $\mathcal{A}$ needs to produce a fake dielet $\mathcal{S}$ which produces a correct verification message $V=[AES_K(C||CB)]_N\oplus(SS||0...0)$ with $SS=$"OK" for some random nonce C and key K corresponding to ID. Since $(SS||0...0)$ is a constant vector, $\mathcal{A}$'s task is reduced to constructing a fake dielet $\mathcal{S}$ which computes the first N bits of ciphertext $AES_K(C||CB)$, where C is a random nonce generated by the trusted smartphone, $CB\in\{CB',...,CB'+T-1\}$, and K and CB' correspond to ID in the server's database. Being able to play both the role of dielet and smartphone in the authentication protocol does not teach $\mathcal{A}$ anything about how to construct such an $\mathcal{S}$ since C will be chosen at random after these authentication protocol interactions. So we can change the security game by replacing $\mathcal{A}^{Auth(.,.,\mathcal{V}_m)}(ID)$ by just $\mathcal{A}(ID)$.

Adversary $\mathcal{A}$ has no oracle access to $[AES_K(.)]_N$, therefore, if restricted to t computation steps, $\mathcal{A}$ can distinguish truncated ciphertexts computed by $[AES_K(.)]_N$ from a truly random function with advantage at most $\epsilon_{tAES}(t,0,N)$. If the attacker cannot distinguish $[AES_K(.)]_N$ from a truly random function in t computation steps, then we may change the security game by replacing $[AES_K(.)]_N$ with a truly random function $f(.)$. In this case the attacker's task is to compute the N-bit output of $f(C||CB)$, where C is randomly generated by the trusted smartphone and $CB\in\{CB',...,CB'+T-1\}$. Because the output of a truly random function is uniformly distributed, the probability to guess one correct output is exactly $1/2^N$. Notice that there are T counter blocks which will be accepted by server $\mathcal{V}_m$, so the probability of passing the authentication protocol is $T/2^N$. Summarizing, the probability $\text{IA1}(t,m,\mathcal{A})=1$ is at most $\epsilon_1=\epsilon_{tAES}(t,0,N)+T/2^N$.

4.5.2 IA2 Security

By the definition of initialization protocol $Init(.)$, we know that each secret key is generated independently. Hence, having oracle access to multiple dielets $\{\mathcal{D}_j|j\neq i \text{ and } i,j<m\}$ does not teach adversary $\mathcal{A}$ anything about dielet $\mathcal{D}_i$. Therefore, since the to-be-produced fake dielet $\mathcal{S}$ only needs to output one verification message V for passing one authentication session corresponding to one

specific serial ID, we can replace oracle access to $\mathbb{D}$ with oracle access to just one of the $\mathcal{D}_i$ which serial ID is adopted by $\mathcal{S}$. So, without loss of generality, adversary $\mathcal{A}$ querries $\mathcal{D}_i$ exactly q times and stores a set of q nonce and verification message pairs $\{(C_j, V_j)\}_{j=1}^{q}$, where

$$V_j = [AES_K(C_j||CB_j)]_N \oplus (SS||0\ldots 0) \tag{1}$$

with no counter block CB_j repeating.

See the argument used in proving IA1 security, knowledge of $\mathcal{L}(\mathcal{V}_m)$ only helps in selecting a valid serial ID, in fact, the fake dielet $\mathcal{S}$ should adopt the serial ID of $\mathcal{D}_i$ in order to maximize the probability of successful impersonation (see below). Also, see the proof of IA1 security, playing the role of dielet and smartphone in authentication protocol instances with the server does not help the adversary in constructing $\mathcal{S}$. Summarizing we can change the security game IA2 by replacing $\mathcal{A}^{Auth(.,.,\mathcal{V}_m),\mathbb{D}}(\mathcal{L}(\mathcal{V}_m))$ with $\mathcal{A}^{\mathcal{D}_i}(ID)$.

If in the authentication session the random nonce C sent by the smartphone is equal to one of the C_h, then the fake dielet $\mathcal{S}$ may either send back the verification message V_h or compute some other message V in order to pass authentication. Sending V_h results in a successful impersonation if the counter value CB_h used in producing (C_h, V_h) is in the curent range of acceptable counter values $\{CB', \ldots, CB' + T - 1\}$. Since no CB_j repeats, at most T out of the q pairs (C_j, V_j) was produced with a $CB_j \in \{CB', \ldots, CB' + T - 1\}$. Since random nonce C is selected uniformly out of 2^M bit strings, the probability of C colliding with one of these $\leq T$ nonces C_j is

$$\leq T/2^M. \tag{2}$$

Let us now assume that C does not collide with any of these $\leq T$ nonces C_j. Then as in the proof of IA1 security the task of adversary $\mathcal{A}$ is to generate a correct ciphertext $[AES_K(C||CB)]_N$ for a $CB \in \{CB', \ldots, CB' + T - 1\}$. The difference with IA1 is that $\mathcal{A}$ has collected information about the q ciphertexts in (1) (notice that $\mathcal{A}$ can subtract the known vector $(SS||0\ldots 0)$ with $SS =$"OK"). The probability of generating a correct ciphertext is equal to the probability of winning the following impersonation game:

Impersonation $ImperAuth(t, q, \mathcal{A})$
$\mathcal{A}$ is given knowledge of CB' and T.

1. A random key K is chosen.
2. The adversary $\mathcal{A}$ is given oracle access to the encryption oracle $[AES_K(.)]_N$. After q queries to the oracle and t computation steps, the adversary eventually outputs a fake dielet $\mathcal{S}$. Let $\mathcal{Q}$ denote the set of all oracle queries.
3. Verifier $\mathcal{V}_m$ randomly chooses a C such that $(C||CB') \notin \mathcal{Q}, \ldots, (C||CB' + T - 1) \notin \mathcal{Q}$.
4. The output of the impersonation experiment is defined to be 1 (i.e., the adversary wins the impersonation game) if and only if $[AES_K(C||CB)]_N \leftarrow \mathcal{S}(C)$ for a $CB \in \{CB', \ldots, CB' + T - 1\}$.

We proceed with using the proof technique used for proving an unforgeable Message Authentication Code construction, see THEOREM 4.4 in [35]. Let $\mathcal{A}$ be an adversary and define

$$\epsilon = Pr[ImperAuth(t, q, \mathcal{A}) = 1].$$

We first consider the modified impersonation game $ImperAuth^*$ where $[AES_K(.)]_N$ is replaced by a truly random function $f(.)$ unknown to adversary $\mathcal{A}$ and therefore unknown to $\mathcal{S}$. Since $f(.)$ produces uniformly distributed outputs, $\mathcal{S}$ at best guesses $[AES_K(C||CB)]_N$ for a $CB \in \{CB', \ldots, CB' + T - 1\}$, hence,

$$Pr[ImperAuth^*(t, q, \mathcal{A}) = 1] \leq \frac{T}{2^N}.$$

Given an adversary $\mathcal{A}$ in (the original) $ImperAuth$, we now construct an adversary $\mathcal{B}$ whose goal is to determine whether he has oracle access to function $[AES_K(.)]_N$ or the truly random function $f(.)$; $\mathcal{B}$ is limited to q oracle queries and t computational steps. In the construction below $\mathcal{B}$ emulates $\mathcal{A}$ and observes whether $\mathcal{A}$ succeeds in outputting a successful impersonation for a unused nonce. If so, $\mathcal{B}$ guesses that its oracle is the AES encryption oracle, otherwise $\mathcal{B}$ guesses that its oracle is the random function $f(.)$:

Distinguisher $\mathcal{B}(t, q)$
$\mathcal{B}$ is limited to t computational steps and q queries to an oracle $\mathcal{O}$.

1. Run $\mathcal{A}^{\mathcal{O}}$ to generate a fake dielet $\mathcal{S}$ and record all oracle queries in $\mathcal{Q}$.
2. Select a random $C \leftarrow \{0,1\}^M$ randomly with $(C||CB') \notin \mathcal{Q}, \ldots, (C||CB' + T - 1) \notin \mathcal{Q}$.
3. Compute $y \leftarrow \mathcal{S}(C)$.
4. Query $\mathcal{O}$ with inputs $(C||CB'), \ldots, (C||CB' + T - 1)$. Let $y_1, \ldots, y_T$ be the outputs generated by $\mathcal{O}$.
5. Output 1 if $y \in \{y_1, \ldots, y_T\}$, otherwise output 0.

Notice that if $\mathcal{O} = [AES_K(.)]_N$, then the view of $\mathcal{A}$ executed as a sub-routine by $\mathcal{B}$ is distributed identically to the view of $\mathcal{A}$ in experiment $ImperAuth(t, q, \mathcal{A})$. Furthermore, $\mathcal{B}$ outputs 1 exactly when $\mathcal{A}$ wins in $ImperAuth(t, q, \mathcal{A})$. Therefore,

$$Pr[\mathcal{B}^{[AES_K(.)]_N}(t, q) = 1] = Pr[ImperAuth(t, q) = 1] = \epsilon.$$

Similarly, if $\mathcal{O} = f(.)$, then

$$Pr[\mathcal{B}^{f(.)}(t, q) = 1] = Pr[ImperAuth^*(t, q) = 1] \leq \frac{T}{2^N}.$$

This proves

$$|Pr[\mathcal{B}^{[AES_K(.)]_N}(t, q) = 1] - Pr[\mathcal{B}^{f(.)}(t, q) = 1]| \geq \epsilon - \frac{T}{2^N}.$$

See Def. 4, the probability of distinguishing AES from a truly random function with q queries and t computational steps is $\epsilon_{tAES}(t, q, N)$, so $\epsilon_{tAES}(t, q, N) \geq \epsilon - \frac{T}{2^N}$ and we obtain

$$\epsilon \leq \epsilon_{tAES}(t, q, N) + \frac{T}{2^N}. \tag{3}$$

Since each dielet has a maximum counter block value MAX, the number of useful queries to the dielet is limited by $q \leq MAX$. Combining (2) and (3), gives the desired upper bound on the probability of wining IA2: The probability $\text{IA2}(t, m, \mathcal{A}) = 1$ is at most $\epsilon_{tAES}(t, MAX, N) + \frac{T}{2^N} + \frac{T}{2^M}$.

4.5.3 IA3 Security

A weak-IA3 attacker complements the capabilities of an IA2 attacker in that he can choose to separate a dielet from a legitimate package and glue it as a fake dielet $\mathcal{S}$ on a package of a counterfeit chip. According to the definition of $Sep(.)$, this separation process triggers the passive sensors with probability ρ and we assume that the gluing process is non-invasive and does not trigger additional sensors. So, when engaged next in an authentication protocol a separated dielet produces a verification message

$$V = [AES_K(C||CB)]_N \oplus (SS||0\ldots0),$$

where SS ="OK" with probability $1-\rho$ and $SS \neq$"OK" with probability ρ.

Since sensors are more likely to get triggered than not, $\rho > 1-\rho$ and simply guessing that V corresponds to $SS \neq$"OK" is the most likely hypothesis. Based on V itself, the adversary may do better and may be able to guess (distinguish) whether V corresponds to $SS =$"OK" or $SS \neq$"OK" with probability $> \rho$. Let ϵ_0 be equal to the probability that $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ guesses $SS =$"OK" conditioned on V corresponding to $SS =$"OK". Similarly, let ϵ_1 be equal to the probability that $\mathcal{A}$ guesses $SS \neq$"OK" conditioned on V corresponding to $SS \neq$"OK". We use $\mathcal{A}$ as the adversary in the CPA indistinguishability experiment: Adversary $\mathcal{A}$ wins the CPA indistinguishability experiment (which uses an unbiased bit b with $Pr[b=1] = 1/2$) with probability

$$\epsilon_0/2 + \epsilon_1/2 \leq 1/2 + \epsilon_{cAES}(t, q, N).$$

The probability that $\mathcal{A}$ distinguishes whether V corresponds to $SS =$"OK" or $SS \neq$"OK" (not conditioned on V) is equal to

$$\begin{aligned}(1-\rho)\epsilon_0 + \rho\epsilon_1 &\leq \rho(\epsilon_0 + \epsilon_1) \\ &\leq \rho(1 + 2\epsilon_{cAES}(t, q, N)) \\ &\leq \rho + 2\epsilon_{cAES}(t, q, N).\end{aligned}$$

This implies that $\mathcal{A}$ can only do slightly better than plain hypothesis testing and is able to filter out a separated dielet which has probability

$$\leq (1-\rho) + 2\epsilon_{cAES}(t, q, N)$$

of having $SS =$"OK". By using the same argument as in the analysis of IA2 we may assume that q is limited by MAX. This gives the required upper bound.

We notice that our protocol does not have full IA3 security as this should also resists the unrealistic adversary who can built $\mathcal{S}$ circuitry which can intercept, modify, and resend a message V from a separated dielet: $\mathcal{S}$ simply takes the intercepted V and sends instead

$$V \oplus (\text{"OK"}||0\ldots0) \oplus (GuessSS||0\ldots0)$$

where $GuessSS$ is the guessed status of the sensors in the separated dielet. If the guess is correct, then the new V will pass authentication. The probability of guessing SS is significant as there are only a few sensors. If we wish to also protect against this adversary we may modify our protocol and replace the construction of V by

$$V = [AES_K(C||CB||SS)]_N$$

and let the authentication server verify whether $V = [AES_{K'}(C||CB''||\text{"OK"})]_N$ for a $CB'' \in \{CB', \ldots, CB' + T - 1\}$. If not, different SS can be tried in order to determine which sensors were triggered; as this takes extra computation time (a scarce resource at the authentication server) we did not take this solution as the focus of our paper.

4.5.4 IA4 Security

A variety of successful attacks on embedded AES encryption engines from both academia and industry have been demonstrated in recent years, including power side channels [20, 37, 38], EM side channels [21], differential fault analysis [22,23,39,40] and probing attacks [24,41]. These attacks are all able to extract the secret key out of an AES encryption core in an embedded system allowing an attacker to make a perfect counterfeit of the original dielet and pass future authentication.

In our solution we maximize the cost for one successful impersonation $\sigma = \frac{\beta}{\delta}$ by limiting the number of successful impersonations δ of one compromised serial ID and key pair to MAX, the maximum number of allowed encryption per dielet.

In addition, CTR mode makes it harder to exploit side channel analysis and differential fault analysis, i.e., β increases significantly: An attacker can have at most MAX power traces for each dielet. This is $\ll 2^{13}$ power traces needed in [37] which describes how to break AES-CTR mode encryption using power analysis. This means that in order to attack our dielet an attacker needs power traces with higher signal to noise ratio and more time to brute force the search space. Also CTR mode encryption complicates differential fault analysis by updating the counter value irreversibly. Even in the most powerful differential fault analysis on AES [23] it is necessary to obtain one pair of a fault-free ciphertext and a faulty ciphertext both corresponding to the same plaintext. Therefore, in order to attack our dielet an attacker has to inject faults into the counter block in order to repeat some counter values [42, 43]. For completeness, our solution is as much vulnerable to probing attacks as other solutions; see [3] for a detailed discussion about how an appropriate sensor design can help the SHIELD program.

4.5.5 Suggested Parameters

An appropriate choice of N and M limits the number of bits to be communicated and can save energy otherwise needed by the RF transceiver. E.g. setting $N = 50$, $T = 8$, $M = 50$ and $MAX = 256$ gives $Adv_{tRP}(50, 128, 0) = 0$ for IA1 and $Adv_{tRP}(50, 128, 256) \approx 2^{-56}$ for IA2 and weak-IA3. By assuming $\epsilon_{AES}(t, q)$ is negligible, this yields in the authors' opinion a small enough probability 2^{-47}, 2^{-46} and $(1 - \rho) + 2^{-55}$ of successful IA1, IA2 and weak-IA3 attacks.

E.g., the server may adopt the policy that after a burst B of failed authentication attempts by a smartphone $\mathcal{P}$ for a specific dielet $\mathcal{D}$ the server informs $\mathcal{P}$ to simply notify its user that the dielet $\mathcal{D}$ it is communicating with does not behave normally and that the server will not process any further authentication requests coming from $(\mathcal{P}, \mathcal{D})$ (the server blacklists this pair). Then the overall probability of successful IA1 and IA2 attacks for one smartphone and dielet pair is at most $B \cdot 2^{-47}$ and $B \cdot 2^{-46}$ respectively. For $B = 64$, this means that at most one out of $2^{40} = 1.1 \cdot 10^{12}$ produced fake dielets succeeds making these attacks prohibitively expensive for any economically motivated adversary.

4.6 Performance Improvement

In regards to performance DARPA's example authentication protocol (Fig. 4) suffers from three drawbacks. First, the challenge and the sensor status bits are encrypted separately, which doubles the power consumption of the encryption engine. Second, two full 128-bit ciphertexts have to be transmitted from the dielet to the smartphone through the RF transceiver. This results in a long communication latency between the dielet and the smartphone with a large power consumption. Finally, DARPA's protocol needs two full communication rounds between the smartphone and the remote server which makes the protocol unnecessarily slow due to network latency.

(1) Our solution only needs to encrypt once per authentication request, saving compared to DARPA's protocol half of the power consumption of AES encryption on the dielet.

(2) Our solution transmits fewer bits between the dielet and the smartphone making transmission more efficient. In step (15) in Fig. 5 the dielet sends a 128-bit serial ID (DARPA suggests a 64-bit serial ID; we explain in Sec. 5 why we need 128 bits). In step (16) in Fig. 5 the smartphone replies $L + M$ bits to the dielet to confirm the authentication request. In step (20) in Fig. 5, the dielet sends N bits back to the smartphone. The total number of bits transmitted between the dielet and the smartphone is therefore $128 + M + L + N$. We recommend $L = 30$, $M = 50$ and $N = 50$, which results in only 258 transmitted bits. This is approximately 1.7 times less than the 448 transmitted bits in DARPA's protocol.

(3) Finally, the computation time or transmission time between the dielet and the smartphone is in microseconds, but the network communication latency is on the order of tens of milliseconds [7]. Hence, it is important to minimize the number of communication rounds between the smartphone and the server. Since only one full communication round is needed between the smartphone and the server in our protocol, the entire authentication can complete much faster (SHIELD requires that the authentication protocol finishes in 2 seconds [2]).

4.7 Remarks

4.7.1 Detection of Number of Dielet Readouts

The counter on the dielet allows the server to learn how many times the dielet has been put into authentication mode when the dielet was offline with respect to the authentication server. This information can be used (besides the passive sensor data) as an additional sensor to record suspicious attacking attempts.

Also, as discussed above, our protocol implements a maximum possible counter value MAX after which a dielet will not enter authentication mode any more. This means that even if an IA4 attacker obtains a non-expired pair of a serial ID and cryptographic key, the attacker will only be able to use the pair to authenticate at most MAX counterfeit chips. This demotivates an economically motivated adversary to invest resources in extracting keys from dielets through hardware reverse engineering, imaging etc.

4.7.2 Integration with RFID tags in Supply Chain Management

In current SCM (Supply Chain Management), RFID (Radio Frequency Identification) tags are used as EPCs (Electronic Product Codes) to track products in the supply chain [44]. A dielet inserted in the host package of a chip can besides the required SHIELD functionality also implement EPC functionality. The dielet can use the RF channel to communicate messages to a smartphone or

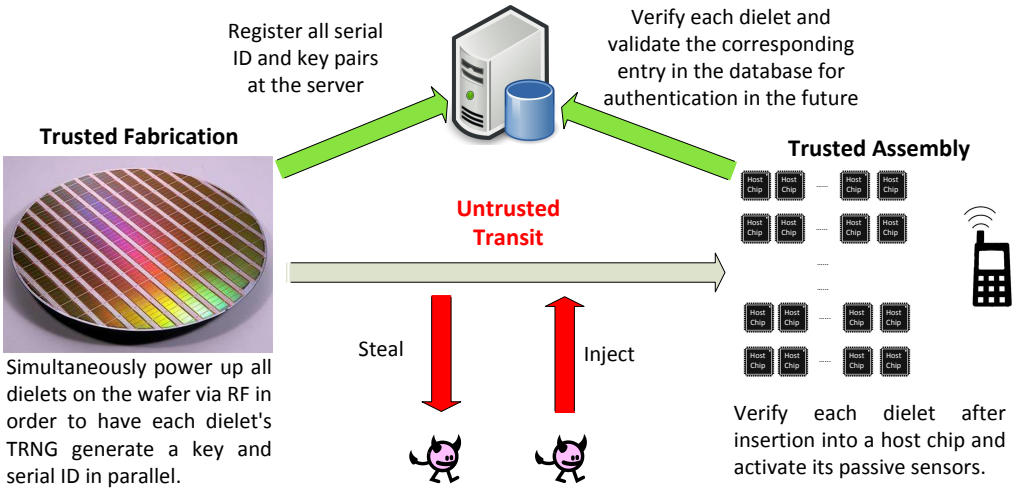

Figure 6: Our proposed initialization process and the adversarial models in initialization.

RFID reader and this allows a seamless integration of SHIELD and EPC functionality in dielets. The same centralized remote server can collect the sensor status bits as well as tracking information of chips in the supply chain.

5 Initialization Protocol

Dielets need to be initialized with their own unique serial IDs and keys, and also the authentication server needs to store a copy of these serial ID and key pairs in its own database. Finally, an initialization process should activate dielets in that their passive sensors will start recording tampering events in their sensor status bits.

Clearly, the passive sensors of a dielet should not be enabled before the dielet is inserted into a host package of a legitimate IC (otherwise, the passive sensors register the insertion as a tampering event). [2] As we have explained in Sec. 3.2, untrusted transit between trusted fabrication and insertion of dielet introduces the risk of stealing and injecting non-validated dielets. Fig. 6 depicts our initialization protocol and attack scenarios.

We propose to let dielets generate their own serial ID and key pairs during a first initialization phase while they are still on the wafer in the trusted dielet fabrication facility. The serial ID and key pairs are uploaded to the server before the dielets leave the trusted fabrication facility. During a second initialization phase each dielet received by the trusted IC assembly facility is verified by our proposed authentication protocol; this detects any malicious dielet injected during untrusted transit between dielet fabrication and assembly. Only if verified a dielet's ID is validated; this prevents an attacker from stealing a not-yet-inserted legitimate dielet, i.e. a dielet with a valid ID, and inserting the dielet into a malicious or counterfeit chip. Once a dielet is inserted, authenticated and validated, the passive sensors of the dielet are activated so that from then onward they can

[2] It is outside the scope of this paper to develop an insertion technology which keeps passive sensors fresh in that they keep on being capable of detecting (with a significant probability) the first future tampering event (the sensor status bits only need to record whether a tamper event has happened and do not need to record how many have happened). Especially, if a sensor is using e.g. photo sensitive material, then this becomes a challenge.

record detected tampering events (in the sensor status bits).

Rather than fusing serial IDs and cryptographic keys directly into each dielet while they are still on the wafer[3], we propose to have each dielet self-generate its key and serial ID *in parallel* by exploiting on-chip randomness by using a TRNG while they are still on the wafer at the trusted dielet fabrication facility.

5.1 True Random Number Generator

A TRNG (True Random Number Generator) is a hardware security primitive, which harnesses the on-chip noise to generate a random number for security applications. A good TRNG design relies on a good entropy source, a decent harvesting mechanism and a postprocessing mechanism [46]. A good entropy source should provide as much entropy as possible, while a decent harvesting mechanism should be able to collect as much entropy as possible from the entropy source. The postprocessing mechanism is not necessary in every TRNG design, but it strengthens the TRNG design and eliminates or reduces the bias in the output bits. E.g., we can implement a von Neumann extractor [47], which is a simple finite state machine for extracting a non-biased true random number from a biased entropy source. The idea is to repeatedly measure two consecutive output bits of the TRNG until a 01 or 10 is measured. A 01 is interpreted as a 0 and a 10 is interpreted as a 1. Since a 01 and 10 are equally likely, their 0 and 1 bit interpretation is un-biased.

We propose to use the SRAM-based TRNG in [48] because it harvests randomness of the noise generated by an area efficient metastable structure which is just an SRAM cell. This means that a dielet only needs one SRAM cell structure to implement a TRNG (which is repeatedly evaluated in order to generate a serial ID together with a cryptographic key).

5.2 Initialization Protocol

Fig. 5 (a) shows the self-generation mode of our proposed initialization protocol: The manufacturer can power up all the dielets in one wafer simultaneously by using a large power RF antenna. This enables the self-generation of a key and a *random* serial ID (which are stored in NVM once generated) in each dielet in parallel. The parallel generation and storage of keys and random serial IDs in the dielets' NVMs takes several microseconds. Once generated they will be uploaded one at a time to the server when the dielets "leave" the wafer (note that the server may in addition associate a unique serial ID in standard format to each random serial ID in its data base). The counter block CB is initialized to zero, and set to one after the self-generation mode has finished.

Once all the dielets on the wafer generated and uploaded their serial IDs and keys, they are ready to leave the trusted dielet fabrication facility. When they arrive at the trusted IC assembly facility, the dielets are inserted into the host package of legitimate ICs and complete the initialization mode of our proposed initialization protocol, see Fig. 5 (b): This mode verifies each dielet received at the IC assembly facility by using our authentication protocol, activates all the passive sensors on the dielets and validates the corresponding entry in database for future authentication outside the trusted environment. The counter block CB is incremented to 2.

[3]One solution is to add extra power lines and access circuitry on the wafer itself, which increases fabrication costs. Another solution is to use RF communication to sequentially write and fuse each unique serial ID and key. This leads to a non-parallelized approach and a simple calculation assuming one million $0.01mm^2$ dielets on a single wafer (dielets are required to fit inside a $0.01mm^2$ area [2]) and assuming a maximum rate of fusing 2500 dielets per second (using high performance equipment [45]) show that initialization may take an impractical 7 minutes via the RF channel.

5.3 Security Analysis

The security of our initialization protocol is reduced to the security of the authentication protocol: The attacker has to break the authentication protocol in order to inject counterfeit/malicious dielets into the supply chain.

5.4 Serial ID Collision

Since a randomly generated serial ID cannot guarantee uniqueness, the length of the serial ID needs to be adapted in order to make the probability of a serial ID collision negligible. Let λ be the probability of a collision among all generated serial IDs, n be the number of bits of a serial ID, and t be the (maximum) number of dielets we plan to produce. From [49], we know that λ, n, and t are related in the following way:

$$t \approx 2^{(n+1)/2} \cdot \sqrt{\ln\left(\frac{1}{1-\lambda}\right)}.$$

If we choose $t = 2^{44.5} = 2.5 \cdot 10^{13}$, and $\lambda = 2^{-40}$, then the size of a serial ID must be $n = 128$ bits. In DARPA's protocol [2], the serial ID has only 64 bits. As explained in Fig. 5, even though the random serial IDs in our protocols require $n = 128$ bits, still only 258 bits are transmitted between the dielet and smartphone in our authentication protocol. This improves over DARPA's authentication protocol which needs to transmit 1.7 times more bits (448 bits).

5.5 Performance

Because we use a true random number generator to allow each dielet to generate its own serial ID and key, instead of writing a unique serial ID and secret key in it, dielets are able to perform this computation in parallel (without any extra equipment). The time to self-generate and store keys and serial IDs in parallel is negligible per wafer and makes our initialization protocol the first practical solution.

6 Implementation

For our protocols we implemented the extra control logic in 32nm technology [50] (including an SRAM-based TRNG with a von Neumann extractor, an 8-bit counter, and a new state machine with four states corresponding to the four modes of the dielet) and made a comparison with the control logic for DARPA's protocol. In our implementation, we consider the interaction with the transceiver, sensor status bits and NVM as primary input/output to our control logic. In our implementation, we used an SRAM-based TRNG which only uses one SRAM cell to harvest run-time physically random noise [48], together with a von Neumann extractor [47] as post-processing circuitry. In order to save area on the dielet, all operations in our implementation are byte-wise, including a compact 8-bit datapath AES-256 encryption-only core based on the architecture in [51], which instantiates two S-box implementations (one for datapath of state and the other one for key expansion) and takes 224 clock cycles to complete one AES-256 encryption. In this AES encryption engine, we used an S-box implementation proposed by Canright in [52], which first changes the basis of the input value from GF(2^8) to a composite field GF(2^8)/GF(2^4)/GF(2^2), and computes all of the operations in this composite field before converting the basis back in the end. Because Canright

did a thorough investigation on S-box implementation, this has been the optimal (smallest) S-box implementation for over ten years. Although we did not implement the other components on the dielet, we demonstrate the complexity the control logic incurred by our protocols by comparing the additional area with the area of AES-256. We notice that the area of AES-256 takes 55% of the allowed area of the dielet ($0.01mm^2$ [2]) in 32nm technology. The control logic of DARPA's authentication protocol costs only 2% area of the dielet, while the control logic of our protocols is 6% giving an area overhead in control logic of only 4%.

Also, the area of an extra 64-bit NVM (for storing a 128-bit random serial ID compared to a 64-bit serial ID in DARPA's protocol) can be estimated by multiplying the cell size of each bit by 64. Because the cell size of NVM varies from $4F^2$ to $22F^2$, where F is the feature size, the area of a 64-bit NVM is negligible ($< 0.01\%$ in 32nm technology) [53].

The passive sensors are mostly deployed as an additional layer above the circuit, so it increases the thickness of the dielet [3]. In addition, another observation is that existing RF transceiver or antenna technology does not fit the SHIELD area requirement at all, because the size of an RF antenna is proportional to the wavelength of the RF signal [54]. Currently, the smallest antenna record is held by two researchers at BIT (Mesra) Ranchi, India developed in 2013 with size of 14mm $\times$ 11mm [55]. Since the size of the smallest antenna is still far away from DARPA's requirement ($0.01mm^2$), the SHIELD dielet should not communicate via conventional RF technology, and the antenna design for SHIELD is still an open question. This statement is also confirmed in DARPA's Call For Proposal [2].

Compared with the area of AES, passive sensors, RF transceiver, possibly a Built In Self Test (BIST) circuitry [56], and original NVM on the dielet, our additional area utilization (4%) is very small.

7 Conclusion

This paper clearly demonstrates the superiority of AES in CTR mode encryption over plain AES encryption. First by taking advantage of the trust requirement of smartphone, we achieve dramatic performance improvements with respect to a 2$\times$ reduction in the number of communication rounds with the server which speeds up authentication by approximately a factor 2 (due to the relatively large network latency). Second, basing the SHIELD authentication protocol on AES in CTR mode results in a 1.7$\times$ reduction in number of bits transmitted between dielet and smartphone and a 2$\times$ reduction in number of required AES encryptions leading to a significant decrease in power consumption. These performance improvements allow a dielet design that meets the heavily constrained SHIELD specifications with respect to area overhead, power consumption, and speed.

However, much more important is the security improvement offered by AES in CTR mode: Using plain AES only offers deterministic symmetric key encryption allowing an adversary to link ciphertexts over time. This leads to the introduced *try-and-check attack* on DARPA's suggested authentication protocol. The attack *nullifies the main projected benefit of SHIELD* since an adversary is able to eliminate any trace/evidence of his activities that can be detected by dielet sensors. It is of crucial importance to make sure security engineers understand when and how to use AES in CTR mode encryption, in particular for a product as important as SHIELD-dielets which is supposed to create a trusted foundation for embedded systems by restoring trust in outsourced IC fabrication and assembly with respect to supply chains that are out of one's own control. Besides preventing the try-and-check attack, AES in CTR mode also offers plenty of other security benefits:

the counter may serve as an additional indicator of suspicious behavior, the counter can be used to limit the lifetime of dielets which in turn demotivates the economically-motivated attackers to perform IA4 (physical invasive) attacks.

As a second contribution we introduce the first secure and practical initialization protocol for dielets. The main insight is to have a two-phase activation of each dielet, which can be used to secure the untrusted transit between a trusted fabrication facility and assembly facility.

Meanwhile, the additional area overhead of our protocol on top of the area overhead of DARPA's authentication protocol (which excludes initialization) is only 4% of the dielet area size, which is small compared with the required area of AES, passive sensors, RF transceiver, possibly BIST circuitry and NVM on the dielet.

Acknowledgment

This project was supported in part by AFOSR MURI under award number FA9550-14-1-0351. The authors would like to thank Prof. Mark M. Tehranipoor and Prof. Domenic Forte for fruitful discussions.

References

[1] SEMI, "White paper: IP infringement causes $4 billion loss to industry annually," http://www.semi.org/en/Press/P043775, 2008.

[2] DARPA, "Supply chain hardware basic concepts and taxonomy of dependable and secure computing," *Microsystems Technology Office/MTO Broad Agency Announcement*, 2014.

[3] D. Shahrjerdi, J. Rajendran, S. Garg, F. Koushanfar, and R. Karri, "Shielding and securing integrated circuits with sensors," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*. IEEE, 2014, pp. 170–174.

[4] AES, NIST, "Advanced Encryption Standard," *Federal Information Processing Standard, FIPS-197*, vol. 12, 2001.

[5] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proceedings of the IEEE*, vol. 67, no. 3, pp. 397–427, 1979.

[6] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 7, pp. 994–1007, 2012.

[7] K. D. Bowers, M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "How to tell if your cloud files are vulnerable to drive crashes," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 501–514.

[8] K. Huang, J. M. Carulli, and Y. Makris, "Counterfeit electronics: A rising threat in the semiconductor manufacturing industry," in *Test Conference (ITC), 2013 IEEE International*. IEEE, 2013, pp. 1–4.

[9] U. Guin, D. DiMase, and M. Tehranipoor, "Counterfeit integrated circuits: detection, avoidance, and the challenges ahead," *Journal of Electronic Testing*, vol. 30, no. 1, pp. 9–23, 2014.

[10] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," 2010.

[11] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 148–160.

[12] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of PUF-based" unclonable" RFID ICs for anti-counterfeiting and security applications," in *RFID, 2008 IEEE International Conference on*. IEEE, 2008, pp. 58–64.

[13] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security." in *USENIX Security*, 2007, pp. 291–306.

[14] G. K. Contreras, M. T. Rahman, and M. Tehranipoor, "Secure split-test for preventing ic piracy by untrusted foundry and assembly," in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 196–203.

[15] R. W. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," Mar. 27 2007, uS Patent 7,195,931.

[16] X. Zhang and M. Tehranipoor, "Design of on-chip lightweight sensors for effective detection of recycled ICs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, no. 5, pp. 1016–1029, 2014.

[17] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 296–310.

[18] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 51–57.

[19] M. Bushnell and V. D. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer Science & Business Media, 2000, vol. 17.

[20] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO 99*. Springer, 1999, pp. 388–397.

[21] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side channel (s)," in *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, 2003, pp. 29–45.

[22] A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh, "A generalized method of differential fault attack against aes cryptosystem," in *Cryptographic Hardware and Embedded Systems-CHES 2006*. Springer, 2006, pp. 91–100.

[23] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*. Springer, 2011, pp. 224–233.

[24] J.-M. Schmidt and C. H. Kim, "A probing attack on aes," in *Information Security Applications*. Springer, 2009, pp. 256–265.

[25] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater, "On a new way to read data from memory," in *Security in Storage Workshop, 2002. Proceedings. First International IEEE*. IEEE, 2002, pp. 65–69.

[26] S. H. Weingart, "Physical security devices for computer subsystems: A survey of attacks and defences," in *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, 2000, pp. 302–317.

[27] B. Yener, "Csci 4974 / 6974 hardware reverse engineering lecture 15: Anti-tamper technologies." [Online]. Available: http://security.cs.rpi.edu/courses/hwre-spring2014/Lecture15_Antitamper.pdf

[28] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: a very compact and a threshold implementation of AES," in *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 69–88.

[29] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Nrepo: normal basis recomputing with permuted operands," in *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 118–123.

[30] H. Lipmaa, D. Wagner, and P. Rogaway, "Comments to NIST concerning AES modes of operation: CTR-mode encryption," 2000.

[31] N. Fact Sheet, "Suite b cryptography," *National Security Agnecy*, 2008.

[32] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," DTIC Document, Tech. Rep., 2001.

[33] R. Housley, "Using AES counter mode with IPsec ESP," *IPsec Working Group, Internet Draft, RSA Laboratories,(Jul. 2002)*, 2003.

[34] ATM Forum Technical Committee, "ATM security specification version 1.0," *af-sec-0100.001, February*, 1999.

[35] J. Katz and Y. Lindell, *Introduction to modern cryptography: principles and protocols*. CRC press, 2007.

[36] S. Gilboa and S. Gueron, "Distinguishing a truncated random permutation from a random function," *arXiv preprint arXiv:1508.00462*, 2015.

[37] J. Jaffe, *A first-order DPA attack against AES in counter mode with unknown initial counter*. Springer, 2007.

[38] S. B. Örs, F. Gürkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an asic aes implementation," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 2. IEEE, 2004, pp. 546–552.

[39] D. Mukhopadhyay, "An improved fault based attack of the advanced encryption standard," in *Progress in Cryptology–AFRICACRYPT 2009*. Springer, 2009, pp. 421–434.

[40] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *Journal of Cryptographic Engineering*, pp. 1–17, 2014.

[41] H. Handschuh, P. Paillier, and J. Stern, "Probing attacks on tamper-resistant devices," in *Cryptographic Hardware and Embedded Systems*. Springer, 1999, pp. 303–315.

[42] J. G. Van Woudenberg, M. F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on*. IEEE, 2011, pp. 91–99.

[43] A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of aes," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*. IEEE, 2012, pp. 7–15.

[44] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID systems and security and privacy implications," in *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, 2003, pp. 454–469.

[45] AMS, "AS3953A datasheet - ams," http://ams.com.

[46] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *Computers, IEEE Transactions on*, vol. 56, no. 1, pp. 109–119, 2007.

[47] J. Von Neumann, "13. various techniques used in connection with random digits," 1951.

[48] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *Computers, IEEE Transactions on*, vol. 58, no. 9, pp. 1198–1210, 2009.

[49] B. Preneel, C. Paar, and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners.* Springer, 2009.

[50] Synopsys, "32/28nm generic library," http://www.synopsys.com.

[51] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and implementation of low-area and low-power AES encryption hardware core," in *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on.* IEEE, 2006, pp. 577–583.

[52] D. Canright, "A very compact s-box for AES," in *Cryptographic Hardware and Embedded Systems–CHES 2005.* Springer, 2005, pp. 441–455.

[53] D. S. Jeong, R. Thomas, R. Katiyar, J. Scott, H. Kohlstedt, A. Petraru, and C. S. Hwang, "Emerging memories: resistive switching mechanisms and current status," *Reports on Progress in Physics*, vol. 75, no. 7, p. 076502, 2012.

[54] V. Chawla and D. S. Ha, "An overview of passive RFID," *Communications Magazine, IEEE*, vol. 45, no. 9, pp. 11–17, 2007.

[55] S. Pal and M. Chakraborty, "Super compact planar microstrip antenna for ultra wide band applications," 2013, patent ID: KOL/814/2013.

[56] E. J. McCluskey, "Built-in self-test techniques," *Design & Test of Computers, IEEE*, vol. 2, no. 2, pp. 21–28, 1985.