LightSource: Ultra Lightweight Clone Detection of RFID Tags from Software Unclonable Responses

Hoda Maleki, Reza Rahaeimehr, and Marten van Dijk

University of Connecticut
{hoda.maleki, reza.rahaeimehr, marten.van_dijk}@uconn.edu

June 10, 2016

Abstract

Radio-Frequency Identification (RFID) tags have been widely used as a low-cost wireless method for detection of counterfeit product injection in supply chains. In order to adequately perform authentication, current RFID monitoring schemes need to either have a persistent online connection between supply chain partners and the back-end database or have a local database on each partner site. A persistent online connection is not guaranteed and local databases on each partner site impose extra cost and security issues. We introduce a new method in which we use 2-3kb Non-Volatile Memory (NVM) in RFID tags themselves to function as a very small "encoded local database". Our method allows us to get rid of local databases and there is no need to have any connection between supply chain partners and the back-end database except when they want to verify products.

We formally define black-box software unclonability and prove our scheme to satisfy this property. To this purpose, we introduce a simple "XOR-ADD" function and prove it is hard to predict its challenge-response behavior if given only one challenge response pair. The XOR-ADD function with control logic can be implemented using at most 170 gates. This implies that our scheme is compatible with the strict power consumption constraints of cheap EPC Class 1 Gen 2 RFIDs.

Contents

1 Introduction

Counterfeit product injection into the supply chain causes important problems for both supply chain partners and final consumers bringing billions of dollars financial losses to people and companies each year. The International Chamber of Commerce expected the value of counterfeit goods globally would exceed 1.7 trillion dollars in 2015 [1]. Making supply chain management secure is a crucially important task.

In order to detect counterfeit product injection into the supply chain Radio-Frequency Identification (RFID) tags have been used as a low-cost wireless identification method. In an RFID-based supply chain each product is equipped/tagged with an RFID tag which has a unique identifier (ID). RFID tag readers allow to record information in the RFID tags themselves and read this information at a later moment in the supply chain. This infrastructure provides an automated process of scanning and tracking products, which in turn facilitates detection of fake RFID tags used by adversaries to tag counterfeit products [2].

The main challenge in designing an RFID based monitoring infrastructure/scheme is how it can detect injection of fake RFID tags. Here we do not aim to protect against physical attacks where the adversary spends time to separate a tag from a legitimate genuine product and applies it to a fake or counterfeit object: Separating an RFID tag from a genuine product while in transit through the supply chain or after arrival at the customer is costly/risky as this means that for a significant amount of time the product needs to be in possession of the attacker in order to perform the separation attack and this significantly increases his exposure of being detected. Therefore, the economically motivated adversary is discouraged to put such an attack in practice for most supply chains. Even though this tag separation and re-apply attack is certainly possible, in general, it does not imply business risk [3, 4].

The challenge is in how to protect against the adversary who is able to clone or counterfeit the RFID tags themselves as such clones can be applied to counterfeit products by the adversary in a safe location for the adversary. Again, we do not aim to protect against physical attacks where the adversary spends a significant amount of time to physically clone a legitimate RFID tag or even software clone a legitimate RFID tag by using side channel analysis (of power consumption, heat map, microscope scanning, etc.) and programming an adversarial RFID tag with the collected information. For the economically motivated adversary physical inspection of (the non-volatile memory of) an RFID tag (which transits a supply chain) in order to produce a malicious clone is too time-consuming and therefore too risky.

The RFID monitoring scheme needs to protect against the adversary who is in possession of a legitimate RFID tag, who may listen to legitimate readers interacting with the legitimate RFID tag, and who may use his own reader to interact with the legitimate RFID tag for a very short time interval in order to collect information with which an adversarial (or adversarially produced) RFID tag can be (quickly) programmed (after which it can be attached/applied to a counterfeit product and injected into the supply chain). This attack allows an adversary to use legitimate RFID tags as black-boxes, i.e., by interacting with and exploiting the wireless communication interface of legitimate RFID tags. Adversarially produced RFID tags are programmed ("software-clone") with the purpose to pass future authentication protocols when in transit through the supply chain. In this paper we call this a *black-box software cloning attack* and we show that black-box software unclonability is realizable in current standardized cheap RFID technology together with an ultra lightweight algebraic trick.

In related work, see section 2 and Table 1, we give a detailed overview of the current state of the art in RFID monitoring scheme design. Most solutions are either not black-box software unclonable or need to be implemented using too many gates thereby violating strict power constraints in existing RFID technology. Only the "XOR scheme" [5] (slightly enhanced) and the "Trace scheme" [6] are black-box software unclonable and allow implementation with existing RFID tag technology.

Generally, each RFID monitoring scheme needs a back-end database to store product data, tag information, and RFID reader events, such as the data that is sent or written into RFID tags and the data that is read from RFID tags. This data is needed to trace and verify a tag/product. Therefore, there must be a persistent online connection between readers and the back-end system. This is not always the case due to limited or no internet connectivity. As an alternative method local databases are used to temporarily store reader's events. These local databases must be integrated into the back-end database. Maintenance cost and security of the local databases are two main disadvantages. Eliminating the need for local databases has been recognized as a central problem in secure RFID monitoring scheme design.

Contributions: In this paper we provide LightSource, a black-box software unclonable RFID monitoring scheme. Our first version is based on implementing a PRNG; at a cost of a larger additional RFID tag circuitry still implementable with existing RFID technology, it outperforms the XOR scheme in that it requires an order less communication between tags, readers, and back-end server, and it outperforms the Trace scheme in that fake tags can be distinguished from legitimate tags (which reduces the cost of having to test or discard legitimate products).

Our final version replaces the PRNG with a one-time pad stored in NVM; it uses a very small number of gates at the cost of needing 2-3kb NVM in RFID tags. The advantage of this second scheme is that it exploits the idea of pushing all the to-be-recorded trace information into a RFID tag's NVM, i.e., we observe that the supply chain partners do not need any local databases, the tags themselves can be thought of as implementing very small-sized local databases. This implies that during usual reader and RFID tag interactions, no communication with the back-end server is needed at all. Only when a supply chain node, in our model just one of the exit nodes, needs to perform an authentication protocol, help from the back-end system is needed. I.e., the stored trace information is transmitted in encoded form to the back-end server who can verify authenticity of the *complete path* (which includes all previous reader interactions) taken by the RFID tag in the supply chain graph. This is the minimal available connectivity required for a RFID monitoring scheme.

Both versions of LightSource use a "XOR-ADD" function which can be implemented using only 170 gates and for which we prove it generates "software unclonable responses". We formally define this new security property and show how it implies black-box software unclonability in our schemes. LightSource stands for ultra lightweight clone detection of RFID tags from "software unclonable responses". It can be used for dynamic supply chains (i.e. partners regularly join and leave the chain), deals with product recalls, and copes with miswriting (by RFID readers).

Organization: In section 2 we explain related work and compare different solutions with LightSource. In section 3 we model RFID monitoring schemes for supply chain management, and we detail the adversarial model. In section 4 we introduce and define "software unclonable responses" and show and prove the "XOR-ADD" trick as a concrete implementation. Based on software unclonable responses we explain our novel approach LightSource for detection of cloned RFID tags in supply chains in section 5, where we also provide implementation details and simulation results. We conclude with section 6.

Table 1: Comparison of RFID tag monitoring schemes

		Crypto based	Key evolving	PUF	XOR [5]	Trace [6]	**LightSource PRNG-based**	**LightSource NVM-based**
HW req.	Circuit overhead (gates)	AES 2400 [8] SHA-1 5500 [9]	180-300	500-1000 [10]	100& 2-3kb NVM	N	1770	170& 2-3kb NVM
	Compatibility with RFID technology	N	Y	Y	Y	Y	Y	Y
Authentication	Black-box software unclonable	Y	N	N [11]	Y	Y	Y	Y
	Distinguish fake from genuine	Y	N	Y†	Y	N	Y	Y
Performance	Online comm with server (bits per RFID per supply chain node)	200-400	250-350	200	2100	150	164	rec.→0*; aut.→<1920
	Reader-Tag communication (bits per RFID per supply chain node)	200-400	250-350	200	2100	112	164	rec.→156; aut.→<1920
	Local partner databases	Y	Y	Y	Y	Y	Y	N*
	Initialization	Fast	Fast	Slow	Fast	Fast	Fast	Slow
Privacy	Track preserving	Y	Y	Poss.‡	Y	Poss.‡	Poss.‡	Poss.‡
	Trace preserving	Y	Y	Y	Y	Y	Y	Y

* Authentication only needs to happen at exit nodes where back-end server communication is required.
‡ By storing pseudonyms in NVM.
* Recording process.
† If we assume the PUF can not be software cloned.

2 Related Work and Comparison

Trace-based anti-counterfeiting solutions were primarily developed for the pharmaceutical industry; Koh et al. [7] suggested to record track and trace data of pharmaceutical product tags to create drug history. Staake et al. [2] discussed the usage of track and trace data in RFID enabled supply chains for counterfeit detection and stressed the attack possibilities when partners are not willing to record or share tracking data and emphasized its negative effects on counterfeit detection solutions. Followed by this initial research, several RFID-based counterfeit detection methods were proposed that can be classified into two categories:

Cryptographic Verification Based Counterfeit Detection: Counterfeit detection based on cryptographic verification aims at making black-box software cloning (see section 3.2 for a precise definition) hard by using embedded cryptographic primitives such as an encryption scheme, (keyed) hash function, PRNG, etc. Authentication is done by sending a random nonce as challenge to a tag, receiving back from the tag the keyed hash (or encrypted value) of the nonce with a shared secret, and verifying the hash (or encryption) based on knowledge of the shared secret. These methods require extra hardware resources which increases the cost of RFID tags [12]; as a rule of thumb every extra 1,000 gates costs 1 cent [13]; a hash function like SHA-1 takes roughly 5,500 gates [9], and hardware implementations of AES-128 take on the order of 2,400 gates [8]; solutions using lightweight cryptographic primitives require > 2000 gates and are studied in [3, 8, 14, 15]; hash-based schemes are studied in [4, 16–19].

Citing "RFID protocols must be lightweight, taking into account the severe constraints imposed on the available power (induced at the antenna), the extremely limited computational capabilities, the small memory size, and the characteristics of the IC design (e.g., the number of gates available for security code)" [20, 21] and "few gates – on the order of hundreds – remain for security functionality" [22] indicate that only very lightweight solutions can successfully be implemented in current RFID technology and that the above above methods require too many gates (> 2000). This explains the crypto-based column in Table 1: the only hurdle to adoption is the lack of compatibility with current RFID technology.

In order to bring authentication circuitry on the RFID side down to at most 2000 gates three approaches have been developed: key-evolving RFID schemes, PUF based RFID schemes, and the XOR operation-based scheme (the middle columns in Table 1). We explain each technique in more detail below:

Key-evolving RFID schemes [23–26] have been introduced by Peris-Lopez et al. [26] and allow ultralightweight cryptography in that the additional circuit overhead is only in the 100s of gates. During authentication the RFID tag computes (1) a response given a challenge using a shared secret and a couple basic bit wise operations and (2) evolves the shared secret to a new one (which is synchronized with the authentication server). To the extend of our knowledge all of the suggested schemes are shown to be broken with respect to full “disclosure attacks” that leak the underlying evolving secret [27–30]; in our framework key-evolving schemes are not black-box software unclonable, see Table 1. The main reason is that none of the suggested schemes proved or legitimized their security with respect to a formal definitional framework.

PUF-based RFID schemes [10,31,32] also have a much lower complexity (100s of gates). RFID tags embed a Physical Unclonable Function (PUF) which upon receiving a challenge creates a response [31]. An ideal PUF is one which implements a random function from challenges to responses which is hard to predict or learn. The back-end authentication server stores a list of challenge-response pairs for each RFID PUF. Each time an RFID tag is being authenticated, the reader sends a challenge which has not been used previously. The tag transmits the corresponding PUF response back to the reader which forwards it to the back-end system. The back-end system compares the received value from the tag against the stored list of challenge-response pairs. If there is a match, the tag is verified. Even though PUF technology seems quite attractive as the PUF circuit overhead is quite minimal, Becker [11] has demonstrated a cloning attack on a commercial PUF-based RFID tag with only 4 protocol executions, which takes less than 200 ms. In other words PUFs do not offer black-box software cloning resistance, see Table 1 (unless an interface for extracting a shared noiseless key, which minimally implements a hash function and error correction with > 2000 gates, is added).

Juels [5] proposes a XOR operation-based scheme where circuit overhead due to cryptographic primitives is eliminated. It implements a challenge-response scheme which make use of a XOR operation: Each tag shares a list of secret triples $(a_i, b_i, c_i)_{i=1}^m$ with the back-end system. Upon receiving a query by a reader, the tag selects the next pseudonym a_i from its list. A genuine reader (who obtains b_i and c_i from the back-end system) authenticates itself by sending b_i. By verifying b_i, the tag will now send the shared secret c_i in order to authenticate itself to the reader. The scheme releases a different pseudonym a_i upon each query in order to protect against a tracking attack; each tag has not one single identity ID but uses a list of identities or pseudonyms a_i which are discarded once having been used, this gives track privacy in that RFID identities remain private so that RFID tags cannot be tracked through the supply chain by an adversary.

Nevertheless, RFID tags can only store a limited amount of secret triples and these can be queried and listened to by an adversary during m reader RFID tag interactions. This allows an adversary to obtain $(a_i, b_i, c_i)_{i=1}^m$; the a_i values can now be used to track the product and this violates track privacy, even worse the triples themselves allow the adversary to program a software clone and this breaks black-box software unclonability.

To solve this issue, Juels [5] introduces the following improvement to prevent a tracking attack: A tag’s pseudonyms are refreshed by legitimate readers; Once an a_i is used, the triple (a_i, b_i, c_i)

is zeroed out and the reader refreshes all (a_j, b_j, c_j) (including the all-zero $(a_i, b_i.c_i)$) by having the RFID tag XOR each triple with a random string d_j received from the reader. During a next reader interaction the new a_{i+1} is used and the protocol continues as described above. The technique eliminates the use of an expensive hash or encryption, however, at the cost of extra communication. As [5] explains track privacy is guaranteed if the adversary can at most listen to $m-1$ interactions. If m interactions are eavesdropped, then a linear combination of the exchanged bit strings can solve for the current triples (a_j, b_j, c_j) and track privacy as well as black-box software unclonability is broken.

In Table 1, we enhance the XOR scheme by making m large enough (personal communication with industry indicates $m = 16$ should be sufficient) and use an irreversible monotonic increasing counter in the RFID tag to limit the allowed number of reader interactions to $m-1$. Notice that this leads to an increased communication cost.

Evolving tag trace based counterfeit detection: Evolving tag trace based counterfeit detection mechanisms have been proposed in [6, 33–36]. The main idea is to create and store a path history of "reader events" for each RFID tag; when exiting the supply chain the back-end system verifies whether the tag went though a valid path which corresponds to the supply chain graph. Reader events are generally stored in local databases that are synchronized with the back-end system. Lehtonen et al. [34] presents probabilistic techniques to enable clone detection in incomplete traces caused by both tag misreading and partners not sharing tag information. By assuming that events generated by cloned tags are rare, such events can be detected by understanding the process that generates legitimate reader events.

Current state-of the-art originated from Zanetti et al. [36] who presents a detection mechanism that is based on verifying the correctness of consecutive shipping and receiving operations using process and location information in tag events. Each time a tag is queried by a reader, an event is created and stored in the reader's local database. This approach does not write to the tag memory which only contains the tag ID. Clone detection is done at the exit point of the supply chain and is defined as a trace that is not consistent with the supply chain graph.

Zanetti et al. [6] improve [36] by using tag memory together with local databases. In this method, each tag stores two values: a random variable (called tail) and a pointer. Whenever an RFID tag reaches a reader, the reader reads all data in the tag, then creates and stores an event in it's local database. Each event includes, the tag ID, a time stamp T, the tail, and the pointer value p. Next the reader randomizes only the symbol in the tail which is pointed at by p and increases the pointer value p. These changes are written to the tag memory. At the exit point all events of all local databases are aggregated in a centralized server (the back-end system). Events related to legitimate tags diverge from those of cloned tags; this is used to decide whether a tag passes authentication or not. This approach detects clone injection, however, one cannot distinguish between a genuine tag and its cloned version. This means that not only the cloned product but also the legitimate product will need to be filtered out, which is costly.

Our Approach: Our first scheme implements the challenge-response authentication of cryptographic verification based counterfeit detection using a PRNG with a new ultra lightweight XOR-ADD trick which generates "software unclonable responses" (see next section). Our lightweight primitive is in this regard information theoretically secure and allows us to prove black-box software cloning resistance. The PRNG only leads to an additional 1600 gates. The XOR-ADD trick costs less than 170 gates. As indicated in the table, at the price of 1770 gates our PRNG-based scheme is at the edge of being compatible with RFID technology. If compared with the XOR scheme we reduce the communication by an order of magnitude, and if compared with the Trace scheme we can reliably distinguish fake from legitimate tags (saving the cost of testing or discarding legitimate products).

The PRNG-based scheme needs server communication during each reader interaction and this requires local partner databases as explained in the introduction. Our second scheme replaces the PRNG by initializing a 2-3kb RFID tag memory with random nonces and this allows us to design an evolving tag trace based counterfeit detection scheme which does not use any local database: Each RFID tag records itself all the events necessary for its authentication and consistency of their complete history of reader interactions throughout the supply chain. Only once at an exit node the authentication protocol is executed and this requires back-end server communication. This implies that supply chain partners do not need to worry about internet connectivity problems. Performing authentication at exit nodes is flexible with respect to when the authentication is done; e.g., someone delivering or installing a product can wait for a couple of minutes till internet connectivity is back.

We notice that any scheme (including ours without sacrificing the no local databases property) can be made to preserve track privacy by complementing solutions with the idea of pseudonyms: Each RFID tag stores, instead of a single ID, a list of random pseudonyms $ID_1, ID_2, \ldots$. When asked to authenticate itself, the RFID tag uses the next (irreversibly) pseudonym in its list which has not yet been used. The back-end server looks up the pseudonym in its database so that it can be connected to the actual ID of the tag. For completeness, trace privacy is concerned with keeping RFID states private as these can otherwise be read out and teach the honest-but-curious supply chain partner data related to the path taken by the RFID tag through the supply chain. In LightSource we use a cheap one-time-pad trick to preserve trace privacy.

3 Background and Adversarial Model

3.1 RFID-Based Supply Chain

A supply chain is a network of partners such as manufacturers, retailers, transporters and distributors that cooperate in the production, delivery and sale of a particular product. Each product enters the supply chain at a *start node*, travels through several other *nodes* and finally sinks at an *exit node* where the product is delivered to the client. The supply chain does not end its life cycle when the product is delivered to the client; a product (e.g. because of misdelivery) can be recalled and can reenter the supply chain at a *recall node*. Conceptually, a supply chain can be represented by a directed graph with several start (source) and several exit (sink) nodes (each supply chain partner may control several nodes).

An RFID-based supply chain is enhanced by RFID technology to promote real-time monitoring of products while traveling through the supply chain. This improves supply chain visibility which aids the targeted product recalls, regulation of production rate, enforcement of safety regulations and counterfeit detection. An RFID-based monitoring scheme consists of the following components:

RFID tags: An RFID tag is a wireless transmitter attached to a product. In general, it consists of a small microchip with an RF antenna, basic functional capabilities and limited memory. Each RFID tag includes a 96-bits Electronic Product Code (EPC) which serves as a unique identifier

(ID) for the physical object to which the tag is attached; it consists of an 8-bit version EPC, a 28-bits manufacturer identifier, a 24-bits product identifier, and a 36-bits item serial number. In this paper we call the 36-bits item serial number the tag ID. RFID tags come in different flavors: RFID tags can be read-only, write-once read-many, or write-many read-many. Based on the powering method, an RFID tag may be passive, i.e. it does not have any power source of its own and derives its power from the RFID reader, active, i.e. it has its own power source (battery) and can initiate communication with reader, or semi-passive, i.e. it has its own power source but does not initiate communication with readers. EPC Class 1 Gen 2 RFID tags are low-cost passive write-many read-many tags that have a minimum memory of 256 bits, with possible extension to more tag memory. EPC Class 1 Gen 2 RFID tags are used in practice in RFID-based supply chains.

RFID reader: RFID readers represent the nodes in a supply chain graph. An RFID reader is a network connected device with an antenna with which the reader is able to communicate with RFID tags over a radio frequency channel and transmit power to RFID tags.

Algorithm 1 Read Command

```
function READ($ID$)
    State = GetState($ID$); /*From the RFID's NVM*/
    Compute on State;
    Return, i.e. transmit, a computed value;
end function
```

Algorithm 2 Write Command

```
function WRITE($ID, x$)
    State = GetState($ID$); /*From the RFID's NVM*/
    Compute on State with $x$;
    Update State acordingly;
end function
```

Readers communicate with an RFID tag with identity ID through read and write interfaces with the general structure as described in Algorithms (1) and (2). It is also possible to have e.g. a combined write and read interface by extending the write function with a final step which returns, i.e. transmit, a computed value. A reader can only communicate with an RFID tag through these read and write commands (other commands can be transmitted to an RFID tag but we do not consider those here as they do not affect the security of our protocols). We notice that wireless transmission between tag and reader is imperfect and can lead to misreading or miswriting respectively. A *misread* happens when the data of a tag is not read by the reader while a *miswrite* happens when a reader is not able to successfully write data to a tag.

Uncompromised readers engage with RFID tags through WRITE(ID,x) in order to record trace information. In our NVM-based scheme we exploit this feature in the extreme where we record in RFID tags unabbreviated/full trace information including reader identity, time of communication, and message authentication code.

Readers have the following secret and identifying information:

- Reader identities are denoted by RID.

- Reader secret keys are denoted by RK.

In our NVM-based scheme a reader with identity RID and secret key RK takes its own version of real time T, computes the message authentication code

$$HMAC = MAC_{RK}(RID||T||ID),$$

computes the pair

$$x = ([RID||T], HMAC),$$

and writes x to the RFID tag. This process is characterized in Algorithm (3).

Algorithm 3 Recording Full Trace Information at RFID Tags

```
function RecordReaderInteraction($RID, ID$)
    $RK$ = GetKey($RID$);
    $T$ = GetLocalTime($RID$);
    $x = ([RID||T], MAC_{RK}(RID||T||ID))$;
    Write($ID, x$);
end function
```

Trusted Server: A back-end system, a.k.a trusted server, constitutes a database DB which contains data related to the tags and their corresponding products. Each reader is connected to the back-end system over a secure channel; we assume that the back-end system shares secret keys with each of the readers.

Each product enters the supply chain at a start node and travels through the supply chain until it arrives at an exit point. We assume start nodes are trusted and initialize RFID tags on behalf of the trusted server; a general format of an initialization protocol from the perspective of a reader is given in Algorithm 4.

Algorithm 4 Initialization Protocol

```
function Initialize($RID, ID$)
    Fuse $ID$ in RFID tag;
    Compute values $x_j$ according to some security parameter $\lambda$; /*May depend on random coin
        flips and $ID$*/
    $T$ = GetLocalTime($RID$);
    Write($ID, x_j$) for all $j$;
    Update $DB$ with $(ID, [RID, T, (x_1, x_2, \ldots)]$; /* By using secure communication with the
        server; $ID$ is now a "valid identity" */
end function
```

Whenever the authenticity of a product with RFID tag needs to be verified, in particular, we always verify authenticity at exit nodes, then the trusted server engages in an authentication protocol with the RFID tag using a reader as intermediary. From a reader's perspective, our protocols have the structure as described in Algorithm 5.

We notice that misreading and miswriting may lead to failing to authenticate a legitimate RFID tag or lead to authenticating a cloned/fake RFID tag. The authentication protocol should minimize

Algorithm 5 Authentication Protocol

```
function Authentication($RID, ID$)
    Get random challenge $x$ from the server;
    Write($ID, x$);
    $y \leftarrow$ Read($ID$);
    Transmit $y$ to the server;
    Server verifies $y$ w.r.t. $x$ and its data base $DB$;
    Server transmits back to $RID$ whether authentication passes;
    Return pass or fail according to server's response;
end function
```

the probabilities of these false negatives and positives, in particular, authentication of a cloned/fake RFID tag should only happen with negligible probability.

The basic functionality of an RFID monitoring scheme is defined by the read and write commands, the recording algorithm, and the initialization and authentication protocols. It applies to static as well as dynamic supply chains (in a static supply chain the supply chain graph is fixed while in a dynamic supply chain the supply chain graph may be dynamically evolve over time, in particular, the partners in the chain can vary from one market opportunity to another).

3.2 Adversarial Model

The adversary's goal is to inject counterfeit products into the supply chain without being detected by the back-end system. We assume it is too risky for the adversary to engage in involved in the following physical attacks with legitimate RFID tags while these are in transit through the supply chain or are at their final customer destination:

Remove and reapply attack: The adversary removes the legitimate tag from a genuine product and reapplies it to a counterfeit product [16, 31].

Side channel attack: The adversary has access to the RFID tag and collects information over side channels or probes the wires of the tag in order to learn the tag's secret data and model it's behavior [37].

The adversary may engage in short interactions with legitimate RFID tags through their read/write interfaces while they are in transit through the supply chain or are at their final customer destination. We assume that the trusted server has implemented an uncompromised back-end system, that the trusted server and readers have shared keys used for secure communication, and that only a small portion of readers are possibly compromised by the adversary. We assume that manufacturing and initialization of RFID tags is trusted implying that all legitimate RFID tags behave according to their functional spec (in particular, the read and write commands are correct) and have valid IDs.

We define black-box software cloning resistance by an adversarial game in which an attacker attempts to exploit his freedom in terms of black-box access to legitimate RFID tags through their read/write interfaces to program a fake RFID to pass authentication in the supply chain. We will explain how the game can be generalized to include adversarial access to a small portion of compromised readers. The game is detailed in Algorithm 6:

Algorithm 6 Black-Box Software Cloning Game

```
 1: function SoftwareCloningGame($\mathcal{A}$)
 2:     For $j \in J$ with $|J| = poly(\lambda)$, let $ID_j$ be such that
 3:         it would pass $\leftarrow$ Authentication($RID_j, ID_j$),
 4:         where $RID_j$ identifies the last legitimate reader with which $ID_j$ had interaction;
 5:     Let $H$ be the history of all communication through the read and write interfaces between
        any RFID tag and any reader;
 6:     (ReadSim(.),WriteSim(.),$RID'$)$\leftarrow$ $\mathcal{A}^{\textsc{Read}(ID_j),\textsc{Write}(ID_j,.)}((RID_j)_{j \in J}, H)$;
 7:     Inject a tag $\mathcal{S}$ with (ReadSim(.),WriteSim(.)) in the supply chain at reader node $RID'$;
 8:     if $\mathcal{S}$ exits the supply chain being authenticated then
 9:         $b = 1$;
10:     else
11:         $b = 0$;
12:     end if
13:     Return $b$;
14: end function
```

- In lines 2, 3, and 4 we intercept $poly(\lambda)$ legitimate (since they would pass authentication) RFID tags with identities ID_j, $j \in J$. The reader with identity RID_j represents the most recent reader with which RFID tag ID_j had an interaction. RID_j represents the node in the supply chain graph where the adversary intercepts ID_j. In line 3 we say that authentication of ID_j with reader RID_j would pass if the authentication protocol were executed (we do not explicitly execute this protocol in the software cloning game as this would unnecessarily restrict the adversary as he may intercept legitimate RFID tags from nodes which do not perform authentication).

- In line 5 we let H denote the history of all communication through the read and write interfaces between any RFID tag and any reader. As these interactions can possibly be listened to by an adversary, we assume these records to be unprotected and public.

- In line 6 the adversary, defined by a probabilistic polynomial time (ppt) algorithm $\mathcal{A}$ outputs a simulator defined by two ppt algorithms ReadSim(.) and WriteSim(.). These two algorithms represent the read/write interface of a fake tag as the algorithms themselves can be hardcoded (or programmed into) an adversarial RFID tag. In order to create the simulator, $\mathcal{A}$ makes use of the $poly(\lambda)$ legitimate tags with knowledge of interaction history H; this is expressed by giving $\mathcal{A}$ oracle access to Read(ID_j) and Write(ID_j, .) and giving $\mathcal{A}$ input H. Here, we note that the legitimate tags have state which is updated on reads/writes. So, the order of read and write commands with which the legitimate tags are being accessed matters. Finally, $\mathcal{A}$ outputs RID' which is used in line 7 to inject the fake RFID tag at node RID'.

The adversary wins if SoftwareCloningGame($\mathcal{A}$) returns $b = 1$. We say the RFID monitoring scheme defined by Read(.), Write(.,.), RecordReaderInteraction(.,.), Initialize(.,.), and Authentication(.,.) with security parameter λ (implicitly used in each of these algorithms) is black-box software cloning resistant if the adversary wins SoftwareCloningGame($\mathcal{A}$) with $negl(\lambda)$ probability. In practice we are not interested in an asymptotic guarantee, therefore in our security analysis we provide a concrete (exponentially small in λ) upper bound on the probability of a winning adversary.

The above defines the hardness of passing authentication with a fake RFID tag. On the flip side, we also need a correctness requirement: Any non-malicious (legitimate) RFID tag should reliably pass authentication whenever being engaged in an instance of the authentication protocol. Here the authentication protocol should be able to deal with a small fraction of reader miswrites and misreads in earlier interactions of the RFID tag with readers (before the current interaction with the reader which executes the authentication protocol). We do not formally define the notion of correctness; we will address correctness separately for our schemes.

Within our adversarial model, the above notions of black-box software cloning resistance and correctness imply that a RFID monitoring scheme satisfying these notions is able to *distinguish legitimate RFID tags from fake RFID tags*. Also we notice that if a RFID tag fails authentication then black-box software cloning resistance and correctness imply that the tag must be an adversarial one which was injected at some node RID' in the supply chain. As in most other schemes LightSource has the additional property that the authentication protocol is able to *determine the injection point of fake RFID tags*.

The software cloning game can be generalized to include adversarial access to a small portion of compromised readers: In lines 2, 3, and 4 we also define a set of compromised readers and we assume in line 5 $\mathcal{A}$ has access to their execution traces as part of history H. We will address compromised readers separately for our schemes.

Finally, since any RFID tag can be destroyed, Denial of Service (DoS) attacks are not part of our adversarial model.

4 Software Unclonable Responses

Both our RFID monitoring schemes are based on a new primitive which we define and give a concrete instantiation of in this section.

Definition 1. *A function $F_k(x)$ outputs software unclonable responses if there exists a ppt algorithm* Gen*(.) such that for any ppt pair $(\mathcal{A}_0, \mathcal{A}_1)$, the probability that* SoftwareUnclRespGame*$(\mathcal{A}_0, \mathcal{A}_1)$ (see Algorithm 7) returns 1 is negl(λ).*

In Algorithm 7 the attacker is allowed to ask for one output/response of function $F_k(.)$ for some random "secret key" k. Based on a response $F_k(x')$ the attacker tries to learn as much as possible about k so that he can build a ppt simulator $\mathcal{S}$ which predicts/simulates $F_k(.)$. The probability that the simulator predicts the correct response for random inputs should be negligible.

Our RFID tags will use this primitive for authentication by generating a new "secret key" and corresponding software unclonable response based on a reader-supplied input/challenge. This restricts an adversary, who only has black-box access to RFID tags, to only obtain one response per key. Based on just this information, the attacker must try to create/program a fake RFID with a simulator which correctly predicts responses in order to pass authentication. We will show that the black-box software unclonable resistance of our schemes reduce to the security of the underlying function F as defined in Definition 1.

There are numerous heavy weight solutions for functions with software unclonable responses based on standard cryptography. The main problem is to design and prove the security of an ultra lightweight solution as this is needed in our supply chain management application. Below we prove

Algorithm 7 Software Unclonable Response Game

```
function SoftwareUnclRespGame($\mathcal{A}_0, \mathcal{A}_1$)
    $k \leftarrow$ Gen($1^\lambda$);
    $x' \in \{0,1\}^\lambda \leftarrow \mathcal{A}_0(1^\lambda)$;
    $\mathcal{S} \leftarrow \mathcal{A}_1(x', F_k(x'))$; /*$\mathcal{S}$ is a ppt algorithm*/
    $x \in \{0,1\}^\lambda$ is a random input;
    if $\mathcal{S}(x) = F_k(x)$ then
        $b = 1$;
    else
        $b = 0$;
    end if
    Return $b$;
end function
```

that the "XOR-ADD" function

$$F_{(k_0,k_1)}(x) = (k_0 + x) \oplus k_1, \text{ for } k_0, k_1, x \in \{0,1\}^\lambda, \tag{1}$$

where $+$ is binary addition with carry (truncated after λ bits) and where $\oplus$ represents XOR, is such a ultra lightweight solution. (Notice that the XOR function $F_k(x) = k \oplus x$ and ADD function $F_k(x) = k + x$ do not offer solutions as k can be extracted from a pair $(x', F_k(x'))$.)

In the remainder of this section, we prove the next theorem. The proof is stand-alone and can be skipped if the reader wants to proceed with the next section which explains our RFID monitoring schemes.

Theorem 1. *For any ppt pair $(\mathcal{A}_0, \mathcal{A}_1)$, the probability that* SoftwareUnclRespGame*$(\mathcal{A}_0, \mathcal{A}_1)$ for $F_{(k_0,k_1)}(x) = (k_0 + x) \oplus k_1$, $k_0, k_1, x \in \{0,1\}^\lambda$, returns 1 is $\leq 0.85^\lambda$.*

Suppose the adversary knows the pair $(x', F_{(k_0,k_1)}(x'))$ and wants to built a simulator which predicts $F_{(k_0,k_1)}(x)$ for random x. We notice that for

$$\begin{aligned} z &= F_{(k_0,k_1)}(x') \oplus F_{(k_0,k_1)}(x), \\ v &= k_0 + x', \\ w &= x - x', \end{aligned}$$

$$z \oplus v = w + v. \tag{2}$$

Also, notice that given x', since k_0 is random, v is random, and since x is random, w is random. These observations can be used to show that predicting $F_{(k_0,k_1)}(x)$ for a randomly selected input x based on $(x', F_{(k_0,k_1)}(x'))$ where F is defined by (1) is equivalent to predicting z in (2) for a randomly selected input w and unknown/random v. This implies that the probability of SoftwareUnclRespGame$(\mathcal{A}_0, \mathcal{A}_1)$ returning 1 is at most the probability of ZWGame$(\mathcal{A})$ (see Algorithm 8) returning 1 maximized over $\mathcal{A}$, which is equal to

$$\begin{aligned} & 2^{-\lambda} \sum_w \max_z Prob_{v \in \{0,1\}^\lambda}[z \oplus v = v + w] \qquad (3) \\ = \;& 2^{-\lambda} \sum_w \max_z \frac{|\{v : z \oplus v = v + w\}|}{2^\lambda}. \end{aligned}$$

We will design an algorithm which given w finds the z which maximizes this probability (this algorithm can be directly translated in an optimal simulator $\mathcal{S}$ in ZWGame($\mathcal{A}$) and SoftwareUnclRespGame($\mathcal{A}_0, \mathcal{A}_1$)).

Algorithm 8 Finding z based on w

```
function ZWGame($\mathcal{A}$)
    $v \in \{0,1\}^\lambda$ is a random input;
    $\mathcal{S} \leftarrow \mathcal{A}(v)$; /*on input $w$, $\mathcal{S}$ finds $z$ which maximizes $|\{v : z \oplus v = v + w\}|$*/
    $w \in \{0,1\}^\lambda$ is a random input;
    if $\mathcal{S}(w) \oplus v = w + v$ then
        $b = 1$;
    else
        $b = 0$;
    end if
    Return $b$;
end function
```

Algorithm 9 computes for input (z, w) the set $\{v : z \oplus v = v + w\}$ represented as either the empty set $\emptyset$ or a string $v \in \{0, 1, *\}$ where $*$ can take on any bit value. So, the cardinality of set $\{v : z \oplus v = v + w\}$ is equal to 2^c where c is the number of $*$ symbols in the output v of the algorithm. The algorithm shortens the initial bit string w and z to smaller substrings of w and z of equal length throughout the recursive calls while producing a representation $v \in \{0, 1, *\}$.

The algorithm distinguishes between the following cases. First, if $|w| = |z| = 1$, then $w = \alpha$ and $z = \beta$ for some $\alpha, \beta \in \{0, 1\}$. In order to satisfy (2) $\beta \oplus v = \alpha + v$. Since $+$ is addition with carry with truncation, $\alpha + v = \alpha \oplus v$. Hence, $\{v : z \oplus v = w + v\} = \{0, 1\}$ which is represented by the string $*$.

Second, if $|w| = |z| \geq 2$, then $w = w'\alpha_1\alpha_0$ and $z = z'\beta_1\beta_0$ for some bit strings w' and z', and bits $\alpha_1, \alpha_0, \beta_1, \beta_0 \in \{0, 1\}$. Similarly, let $v = v'\gamma_1\gamma_0$. Then, $z \oplus v = v + w$ is equivalent to

$$\begin{aligned} \beta_0 \oplus \gamma_0 &= \alpha_0 \oplus \gamma_0 \text{ and} && (4) \\ (z'\beta_1) \oplus (v'\gamma_1) &= (w'\alpha_1) + (v'\gamma_1) + (0^{|w'|}(\alpha_0 \cdot \gamma_0)), && (5) \end{aligned}$$

where $\alpha_0 \cdot \gamma_0$ represents bit multiplication and is equal to the carry of $\alpha_0 + \gamma_0$. Equation (4) is the projection of $z \oplus v = v + w$ on the first bit and equation (5) is the projection of $z \oplus v = v + w$ on the remaining bits. If $\beta_0 \neq \alpha_0$, then (4) cannot be satisfied and $\{v : z \oplus v = v + w\} = \emptyset$. If $\beta_0 = \alpha_0$, then we distinguish the cases $\beta_0 = 0$ and $\beta_0 = 1$.

For $\beta_0 = \alpha_0 = 0$, any γ_0 solves (4) and equation (5) simplifies to

$$(z'\beta_1) \oplus (v'\gamma_1) = (w'\alpha_1) + (v'\gamma_1).$$

This means that application of SolutionSpace($z'\beta_1, w'\alpha_1$) yields a representation $v \in \{0, 1, *\}$ of all the solutions for (5). Concatenating v with the wild card $*$ (expressing that γ_0 can be any bit value) in line 17 gives a representation of $\{v : z \oplus v = v + w\}$.

For $\beta_0 = \alpha_0 = 1$, any γ_0 solves (4) and equation (5) simplifies to

$$(z'\beta_1) \oplus (v'\gamma_1) = (w'\alpha_1) + (v'\gamma_1) + (0^{|w'|}\gamma_0). \tag{6}$$

Algorithm 9 Outputs the set $\{v : z \oplus v = v + w\}$

```
function SolutionSpace($z, w$) /* defined for $|z| = |w| \geq 1$ */
    if $|w| = |z| = 1$ then
        $\alpha = w$; $\beta = z$;
        if $\alpha \neq \beta$ then
            Return $\emptyset$;
        else
            Return $*$;
        end if
    else/* $n = |w| = |z| \geq 2$ */
        $w'\alpha_1\alpha_0 = w$ with $\alpha_1, \alpha_0 \in \{0, 1\}$;
        $z'\beta_1\beta_0 = w$ with $\beta_1, \beta_0 \in \{0, 1\}$;
        if $\alpha_0 \neq \beta_0$ then Return $\emptyset$;
        else/* $\alpha_0 = \beta_0$ */
            if $\beta_0 = 0$ then
                $z = z'\beta_1$; $w = w'\alpha_1$;
                $v =$SolutionSpace($z, w$);
                Return $v*$;
            else/*$\beta_0 = 1$*/
                $z = z'\beta_1$; $w = w'\alpha_1 + 0^{|w'|}(\beta_1 \oplus \alpha_1)$;
                $v =$SolutionSpace($z, w$);
                Return $v(\beta_1 \oplus \alpha_1)$;
            end if
        end if
    end if
end function
```

The projection of (6) is equivalent to the equation $\beta_1 \oplus \gamma_1 = \alpha_1 \oplus \gamma_1 \oplus \gamma_0$, i.e.,

$$\gamma_0 = \beta_1 \oplus \alpha_1.$$

Substituting this in (6) gives the equivalent equation

$$(z'\beta_1) \oplus (v'\gamma_1) = ((w'\alpha_1) + (0^{|w'|}(\beta_1 \oplus \alpha_1))) + (v'\gamma_1).$$

This means that the application of

$$\textsc{SolutionSpace}(z'\beta_1, (w'\alpha_1) + (0^{|w'|}(\beta_1 \oplus \alpha_1)))$$

yields a representation $v \in \{0, 1, *\}$ of all the solutions for (5). Concatenating v with $\gamma_0 = \beta_1 \oplus \alpha_1$ in line 21 gives a representation of $\{v : z \oplus v = v + w\}$.

As an invariant of algorithm 9 one can prove by induction in the length $\lambda = |z| \geq 1$ that, if $\{v : z \oplus v = v + w\} \neq \emptyset$, then

$$|\{v : z \oplus v = v + w\}| = 2^{\lambda - wt(\bar{z})}, \tag{7}$$

where $wt(\bar{z})$ is the Hamming weight of $\bar{z}$ and $z = \alpha'\bar{z}$ with $\alpha' \in \{0, 1\}$. Maximizing the probability in (3) given w is therefore equivalent to finding, given w, the minimal $wt(\bar{z})$ among all z for which $\{v : z \oplus v = v + w\} \neq \emptyset$.

Let $w = w'01^j0^a$ with $a \geq 0$ and $j \geq 1$. We distinguish two cases: $j = 1$ and $j \geq 2$. If $j = 1$, then $w = w'010^a$. By applying lines 12-17 a times and applying lines 18-19 the $(a+1)$th time for $\alpha_0 = \beta_0 = 1$ and $\alpha_1 = 0$ shows that z must have the form $z = z'\beta_1 10^a$ otherwise no solution for $z \oplus v = v + w$ exists. The recursive call in line 20 uses a new $z = z'\beta_1$ and $w = w'\beta_1$. If $\beta_1 = 1$, then the new w equals $w = w'1$ and $wt(\bar{z}) = 2 + wt(\bar{z'})$ if $|w'| = |z'| \geq 1$, and $wt(\bar{z}) = 1$ if $|w'| = |z'| = 0$. If $\beta_0 = 0$, then the new w equals $w = w'0$ and $wt(\bar{z}) = 1 + wt(\bar{z'})$ if $|w'| = |z'| \geq 1$, and $wt(\bar{z}) = 1$ if $|w'| = |z'| = 0$.

If $j \geq 2$, then $w = w'01^i110^a$ for some $i \geq 0$. By applying lines 12-17 a times and applying lines 18-19 the $(a+1)$th time for $\alpha_0 = \beta_0 = 1$ and $\alpha_1 = 1$ shows that z must have the form $z = z'\beta_1 10^a$ for some bit β_1 otherwise no solution for $z \oplus v = v + w$ exists. The recursive call in line 20 uses a new $z = z'\beta_1$ and a new $w = w'01^i1 + 0^{|w'|+1+i}(\beta_1 \oplus 1)$. If $\beta_1 = 1$, then the new w equals $w = w'01^i1$ and $wt(\bar{z}) = 2 + wt(\bar{z'})$ if $|w'| = |z'| \geq 1$, and $wt(\bar{z}) = 1$ if $|w'| = |z'| = 0$. If $\beta_0 = 0$, then the new w equals $w = w'10^{i+1}$ and $wt(\bar{z}) = 1 + wt(\bar{z'})$ if $|w'| = |z'| \geq 1$, and $wt(\bar{z}) = 1$ if $|w'| = |z'| = 0$.

For both cases, one can show that $wt(\bar{z})$ is minimized for $\beta_0 = 0$ and $z = z'010^a$. Given the above analysis the number of ones in $\bar{z}$ as a function $f(.)$ of w is equal to $f(w'01^j0^a) = 1 + f(w'0)$ for $j = 1$ and $f(w'01^j0^a) = 1 + f(w'10^{j-1})$ for $j \geq 2$. By using similar arguments we can also derive $f(1^j0^a) = 0$ for $j = 1$ and $f(1^j0^a) = 1$ for $j \geq 1$. Since $a \geq 0$ is arbitrary, we obtain the recursion $f(w'01^j0^a) = 1 + f(w')$ for $j = 1$ and $f(w'01^j0^a) = 1 + f(w'1)$ for $j \geq 2$. Since $f(w') \leq f(w'1)$ (the recursion shows that if $w' = w''1$ then $f(w') = f(w''1)$, and if $w' = w''0$ then $f(w') = f(w'') = f(w''01) - 1 = f(w'1) - 1$), we get the inequality

$$f(w'01^j0^a) \geq 1 + f(w') \text{ for } j \geq 1$$

with $f(1^j0^a) = 0$ for $j = 1$ and $f(1^j0^a) = 1$ for $j \geq 1$. So, the minimal weight z as a function of w is at least the number of maximal length substrings 1^j0 with $j \geq 1$ in $\bar{w}$. (Notice that this implies the minimal weight is at most $\lambda/2$.) For completeness, algorithm 10 computes the z of minimal weight given w; this algorithm will be used by the adversary to construct a simulator.

Algorithm 10 Computing a $\bar{z}$ with $\{v : z \oplus v = v + w\} \neq \emptyset$ of minimal weight

```
function MinimalWeightSol($w$)
    if $w = 1^j 0^a$ for some $j \geq 1$ and $a \geq 0$ then
        if $j = 1$ then
            Return $0^a$;
        else
            Return $0^{j-2}10^a$;
        end if
    end if
    if $w = w'01^j 0^a$ for some $j \geq 1$ and $a \geq 0$ then
        if $j = 1$ then
            $z =$MinimalWeightSol($w'$);
        else
            $z =$MinimalWeightSol($w'1$);
        end if
        Return $z0^j 10^a$;
    end if
end function
```

Let W_h, $h \leq \lambda/2$, be the number of bit strings $\bar{w} \in \{0,1\}^{\lambda-1}$ of the form

$$\bar{w} = 0^{a_0} 1^{1+a_1} 0^{1+a_2} \dots 1^{1+a_{2h-1}} 0^{a_{2h}}$$

with $|w| = \lambda$ and $a_i \geq 0$ for all i. The previous analysis shows that for $\bar{w} \in W_h$, $wt(\bar{z}) \geq h$. Together with (7) we obtain

$$2^{-\lambda} \sum_{w} \max_{z} \frac{|\{v : z \oplus v = v + w\}|)}{2^{\lambda}} \leq 2^{-(\lambda-1)} \sum_{h=0}^{\lambda/2} |W_h| 2^{-h}.$$

We notice that the mapping from $\bar{w}$ in W_h to

$$0^{a_0} 10^{a_1} 10^{a_2} 1 \dots 0^{a_{2h-1}} 10^{a_{2h}} \in \{0,1\}^{\lambda}$$

is a bijective mapping from W_h to strings in $\{0,1\}^{\lambda}$ with exactly $2h$ ones. This proves $W_h = \binom{\lambda}{2h}$ and

$$\begin{aligned} 2^{-(\lambda-1)} \sum_{h=0}^{\lambda/2} |W_h| 2^{-h} &= 2^{-(\lambda-1)} \sum_{h=0}^{\lambda/2} \binom{\lambda}{2h} 2^{-h} \\ &= 2^{-\lambda} \left(\sum_{h=0}^{\lambda} \binom{\lambda}{h} \sqrt{2}^{-h} + \sum_{h=0}^{\lambda} \binom{\lambda}{h} (-\sqrt{2})^{-h} \right) \\ &= \left(\frac{1 + \frac{1}{\sqrt{2}}}{2} \right)^{\lambda} + \left(\frac{1 - \frac{1}{\sqrt{2}}}{2} \right)^{\lambda} \approx 0.85^{\lambda}. \end{aligned}$$

This proves the main bound in the theorem.

5 LightSource

In this section we present the PRNG based and the NVM based flavors of LightSource.

5.1 PRNG-Based Flavor

Algorithm 11 PRNG-Based Initialization Protocol

```
function INITIALIZE($RID, ID$)
    Choose random PRNG seed $S$;
    Fuse $ID$ and store seed $S$ in RFID tag;
    Initialize $ctr = 0$;
    $T$ = GetLocalTime($RID$);
    Update $DB$ with $(ID, [RID, T, S])$;
end function
```

Algorithm 12 PRNG-Based Authentication Protocol

```
function AUTHENTICATION($RID, ID$)
    Get random challenge $x$ from the server;
    $(ID, cnt, z)$ ←WRITEREAD($ID, x$);
    Transmit $(ID, cnt, z)$ to the server;
    Server looks up $ID$ in $DB$ and retrieves $(ID, cnt', S')$;
    if $cnt \leq cnt' + \tau$ then /* threshold $\tau \geq 0$ */
        Return fail;
    else
        Forward $S'$ using PRNG over $cnt - cnt'$ iterations;
        $(k'_0, k'_1) = S'$;
        if $z = (k'_0 + x) \oplus k'_1$ then
            $cnt' = cnt$;
            Return pass;
        else
            Return fail;
        end if
    end if
end function
```

The initialization protocol for the RFID tag monitoring scheme is given in algorithm 11 where each RFID tag obtains a unique initial seed and identity (in our protocols and algorithms the instruction $\hat{S}$ =PRNG(S) means that the PRNG updates its state to a new state $\hat{S}$ based on a current state S and for simplicity we assume that $\hat{S}$ also represents the random bit string produced by the PRNG). The authentication protocol is described in algorithm 12 and uses the write-and-read command of algorithm 13 to implement a challenge-response protocol. The RFID tag applies the XOR-ADD function to a random challenge x. Since the challenge x is random and the key (k_0, k_1) cannot be distinguished from random because it is extracted from the output of the PRNG, we may apply Theorem 1 and conclude that the XOR-ADD function is used in our protocols generate

Algorithm 13 PRNG-Based Write and Read Command

```
function WriteRead($ID, x$)
    $(S, cnt)$ =GetState($ID$); /* current state PRNG and counter */
    $S$ =PRNG($S$); /* PRNG computes a new pseudo random state */
    $cnt$++;
    $(k_0, k_1) = S$;
    Return $(ID, cnt, (k_0 + x) \oplus k_1)$;
end function
```

software unclonable responses with probability $\geq 1 - 0.85^{\lambda} = 1 - 2^{-15}$. By assuming that the probability of being able to distinguish the PRNG output from true random is very small, we can show that an adversary can win the black-box cloning game with probability at most 2^{-15} (as each challenge is random with respect to previous ones, recorded histories of interactions cannot help the adversary). As an example, a most successful strategy for an attacker is to:

1. Intercept a legitimate tag and challenge it with some string x' (based on $\mathcal{A}_0$) using the write and read command. This teaches $(ID, cnt, (k_0 + x') \oplus k_1)$.

2. Let the legitimate tag proceed through the supply chain. When challenged next during an authentication protocol with a reader, $cnt \geq cnt' + 2$ at the start of the authentication protocol. If the threshold τ is set large enough in order to accommodate for misreads and miswrites, then the legitimate challenge will still authenticate.

3. The attacker can build a simulator using x' and $(ID, cnt, (k_0 + x') \oplus k_1)$ (based on $\mathcal{A}_1$) and use this to program a fake tag.

4. The fake tag is attached to a counterfeit product and inserted just before a (in our model trusted) exit point. If the simulator predict the correct response during the authentication at the exit point, then the attacker has won the black-box software cloning game.

In practice, the probability of winning is small enough and discourages an economically motivated adversary from trying to physically interact with on average $2^{15} \approx 32000$ legitimate RFID tags before one can be successfully cloned. Also, notice that only after participation in an authentication protocol at an exit node, the adversary will know whether his counterfeit product with cloned tag passes authentication – on average he will lose out on about 31999 illegally inserted products before one successfully exits the supply chain.

5.2 NVM-Based Flavor

Instead of using a PRNG, we initialize RFID tag memory with a pseudo random string $(k_0^j, k_1^j, k_2^j)_{j=0}^{n-1}$ stored in $M_0, M_1, ..., M_{n-1}$ (each M_j stores a triple (k_0^j, k_1^j, k_2^j)) together with a pointer p set to index 0 and mode bit $s = 1$ set to "recording mode": Algorithm 14 describes the initialization; it uses the write command executing lines 2-9 (after fabrication $s = 0$ by default) as detailed in algorithm 15.

After initialization, the write command can only execute line 2 with lines 11-16. Notice that the pseudo random bit strings (k_0, k_1, k_2) stored in M_p during initialization is used to (a) execute the XOR-ADD function with (k_0, k_1) for the second half of the input x_1 and store the result back

Algorithm 14 NVM-Based Initialization Protocol

```
function INITIALIZE($RID, ID$)
    Fuse $ID$ in RFID tag;
    Random seed $S$;
    $(k_0^j, k_1^j, k_2^j)_{j=0}^{n-1} \leftarrow$PRNG($S$);
    $T$ = GetLocalTime($RID$);
    WRITE($ID, (k_0^j, k_1^j, k_2^j)$) for $j = 0$ to $j = n - 1$;
    Update $DB$ with $(ID, [RID, T, S])$;
end function
```

into M_p, and (b) use k_2 as a one-time pad to obfuscate the first half of the input x_0 and store the result back into M_p (left-over bits in M_p are reset back to 0 so that no bits of (k_0, k_1, k_2) are explicitly revealed). The recording algorithm 3 stays as is: during recording reader events $x = (x_0, x_1) = ([RID||T], MAC_{RK}(RID||T||ID))$ are written to the RFID tag.

Algorithm 15 NVM-Based Write Command

```
function WRITE($ID, x$)
    $(M_0, M_1, ..., M_{n-1}, p, s)$ =GetState($ID$);
    if $s = 0$ then /* Initialization mode */
        $M_p = x$;
        $p + +$;
        if $p = n$ then
            $p = 0$;
            $s = 1$;
        end if
    else/* Recording mode */
        $(k_0, k_1, k_2) = M_p$;
        if $p < n$ then
            $(x_0, x_1) = x$; /* $x_0$ represents $RID$ and $T$, $x_1$ represents their MAC */
            $M_p = ((k_0 + x_1) \oplus k_1, x_0 \oplus k_2, 0)$;
            $p + +$;
        end if
    end if
end function
```

The authentication protocol depicted in algorithm 16 uses the read command detailed in algorithm 17. The read command only returns the content of the RFID tag's memory $M_0, M_1, ..., M_{p-1}$ up to the index indicated by its pointer p. This means that during a black-box cloning attack none of the secret information $(k_0^j, k_1^j, k_2^j)_{j=p}^{n-1}$ stored in $M_p, \ldots, M_{n-1}$ leaks.

We consider that the adversary can use the history of interactions H to learn a list of pairs

$$x = (x_0, x_1) = ([RID||T], MAC_{RK}(RID||T||ID))$$

for various RID and times T that were used when written to his intercepted legitimate tags ID_j, $j \in J$. By reading out ID_j, $j \in J$, the adversary learns the information

$$((x_0, x_1), ((k_0 + x_1) \oplus k_1, x_0 \oplus k_2))$$

for all the memory entries stored in each of the tags during the recording protocol. This teaches

$$(x_1, (k_0 + x_1) \oplus k_1) \text{ with } x_0 \text{ and } k_2. \tag{8}$$

At first glance, this seems that the attacker is able to use H to choose (in $\mathcal{A}_0$) a "good" candidate x'_1. However, MAC values are related to both the tag ID and the reader RID which limits the adversaries choice to only the MACs associated to cloned tag ID. If such an old MAC is used, then since it includes the timestamp, the time sequence will not be consistent which is detected by the server in the authentication protocol. Therefore, we can use the same $\lambda = 64$ and the same security analysis of the PRNG-based solution to conclude the same security level.

Algorithm 16 NVM-Based Authentication Protocol

```
 1: function AUTHENTICATION($RID, ID$)
 2:     RECORDREADERINTERACTION($RID, ID$);
 3:     $(p, M_0, M_1, ..., M_{p-1})$ =READ($ID$);
 4:     Server obtains $(ID, [RID', T, S])$ from $DB$;
 5:     Server: $(k_0^j, k_1^j, k_2^j)_{j=0}^{p-1}$ ←PRNG($S$);
 6:     for j=0..p-1 do
 7:         Server: $(z_j, y_j, .) = M_j$;
 8:         Server: $(RID_j, T_j) = y_j \oplus k_2^j$ and looks up $RK_j$;
 9:         Server: $b_j \leftarrow (z_j \equiv (MAC_{RK_j}(RID_j||T_j||ID) + k_0^j) \oplus k_1^j)$;
10:     end for
11:     if $(b_j \equiv False)$ then
12:         Try again in line 2; /* The reader engaging in the authentication protocol should not
            suffer from any misread or miswrite */
13:     end if
14:     Server: Checks whether the sequence of $T_j$ is ascending for $b_j \equiv True$;
15:     Server: Let $m$ be the number of missing nodes and let $e$ be the number of $b_j \equiv False$;
16:     if the above checks pass and $m + e$ is "small enough" then
17:         Return Accept
18:     else
19:         Return Reject
20:     end if
21: end function
```

Algorithm 17 NVM-Based Read Command

```
function READ($ID$)
    $(M_0, M_1, ..., M_{n-1}, p, s)$ =GetState($ID$);
    Return $(p, M_0, M_1, ..., M_{p-1})$;
end function
```

5.3 Implementation and Simulation

Figure 1 shows the main design features of LightSource. The gray diagonal blocks show the additional modules for the NVM-based flavor and the blocks with solid color show the extra modules for PRNG-based flavor.

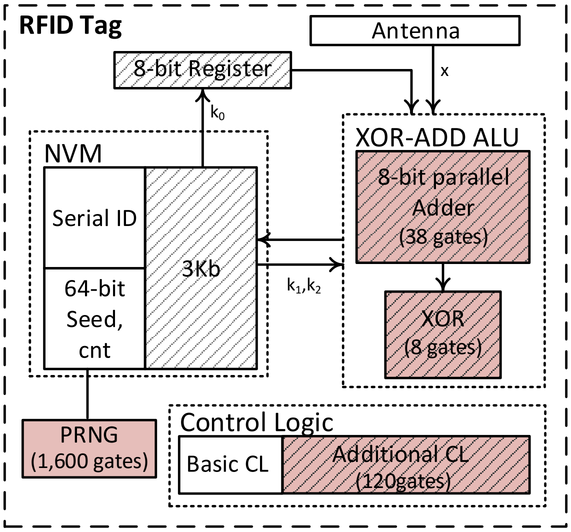

Figure 1: LightSource architecture: shaded blocks indicate the additional hardware in the RFID for LightSource scheme.

NVM-based scheme: The XOR-ADD function is implemented using a 38 gates for 8-bit parallel adder and 8 gates XORs; each message is split into 8-bit blocks and the XOR-ADDing process is repeated $\lambda/8$ times. The control logic is 120 gates: 53 gates for an 11-bit adder, 36 gates for an 11-bit comparator, 22 gates for two 11-bit multiplexers, and 9 other gates. For the write command, each process demands three clock cycles thus in total we need $3\lambda/8$ clock cycles to calculate the result data; for $\lambda = 64$ this takes less than 24 ns. In addition, we use 3kb RFID tag NVM as well as an 11-bit register.

Communication during recording takes 156 bits: a 36-bit tag ID; a 36-bit reader ID, 20-bit time stamp (valid for a 1/2 min clock for 1 year after which the reader key needs to be refreshed and clock needs to be reset), and a 64-bit MAC. Communication during authentication takes < 1920 bits: a 96-bits EPC, and up to 16 reader ID (personal communication with industry indicates the length of a supply chain path is ≤ 16), time stamp, MAC combinations taking $16 \cdot 120 = 1920$ bits. Since 1920 bits seems a lot, we simulated the NVM-based scheme by implementing initialization and authentication protocols in C# using Microsoft SQL Server Express 2012 on a standard laptop (with Windows 10 OS, CPU Core i7-4712HQ, 16GB RAM, and 1TB 5400rpm HDD). Updating the server database for 10^7 RFID tags during initialization happened at 1538 RFIDs per second of which 91.2% of the time data was inserted into the database. Authenticating 10^7 RFID tags happened at 6495 RFIDs per second of which 99% of the time data was retrieved from the database. Concluding, even with a standard laptop's HW, initialization gives only 10% overhead and authentication gives only 1% overhead due to our scheme on top of standard database operations.

PRNG-based scheme: We suggest to use a PRNG which has passed the NIST randomness test suite in addition to Gen2 standard's statistical tests, see [38,39]. The PRNG as described in [39] has a 64 bits seed producing 32 bits pseudo random numbers per cycle and can be implemented in hardware with less than 1600-gates. Besides the PRNG, we assume RFID tags have a monotonic counter and implement our XOR-ADD function. This gives 38 gates for the XOR-ADD ALU and

120 gates for the control logic. Reader interaction requires 224 bits: a 96-bits EPC, 64-bit MAC as challenge, and a 64 bits response.

6 Conclusion

We improved over state-of-the-art RFID monitoring schemes by solving the problem of how to eliminate the need for local partner databases. Based on an ultra lightweight XOR-ADD function with a new provable secure property, our schemes are black-box software unclonable and can be implemented using current RFID technology.

References

[1] F. Economics, "Estimating the global economic and social impacts of counterfeiting and piracy," *A report commissioned by business action to stop counterfeiting and piracy (BASCAP), An ICC Initiative*, 2011.

[2] T. Staake, F. Thiesse, and E. Fleisch, "Extending the epc network: the potential of rfid in anti-counterfeiting," in *Proceedings of the 2005 ACM symposium on Applied computing.* ACM, 2005, pp. 1607–1612.

[3] B. Toiruul and K. Lee, "An advanced mutual-authentication algorithm using aes for rfid systems," *International Journal of Computer Science and Network Security*, vol. 6, no. 9B, pp. 156–162, 2006.

[4] J. Yang, J. Park, H. Lee, K. Ren, and K. Kim, "Mutual authentication protocol for low-cost rfid," 2005.

[5] A. Juels, "Minimalist cryptography for low-cost rfid tags," in *Security in Communication Networks.* Springer, 2005, pp. 149–164.

[6] D. Zanetti, S. Capkun, and A. Juels, "Tailing rfid tags for clone detection." in *NDSS*, 2013.

[7] R. Koh, E. W. Schuster, I. Chackrabarti, and A. Bellman, "Securing the pharmaceutical supply chain," *White Paper, Auto-ID Labs, Massachusetts Institute of Technology*, pp. 1–19, 2003.

[8] 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.

[9] M. O'Neill *et al.*, "Low-cost sha-1 hash function architecture for rfid tags," *RFIDSec*, vol. 8, pp. 41–51, 2008.

[10] P. Tuyls and L. Batina, "Rfid-tags for anti-counterfeiting," in *Topics in cryptology–CT-RSA 2006.* Springer, 2006, pp. 115–131.

[11] G. T. Becker, "The gap between promise and reality: On the insecurity of xor arbiter pufs," in *Cryptographic Hardware and Embedded Systems–CHES 2015.* Springer, 2015, pp. 535–555.

[12] S. Spiekermann and S. Evdokimov, "Privacy enhancing technologies for rfid-a critical state-of-the-art report," *IEEE Security and Privacy*, vol. 7, no. 2, pp. 56–62, 2009.

[13] S. A. Weis, "Security and privacy in radio-frequency identification devices," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[14] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "Aes implementation on a grain of sand," *IEE Proceedings-Information Security*, vol. 152, no. 1, pp. 13–20, 2005.

[15] M. Feldhofer, "An authentication protocol in a security layer for rfid smart tags," in *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, vol. 2. IEEE, 2004, pp. 759–762.

[16] J.-S. Cho, S.-S. Yeo, and S. K. Kim, "Securing against brute-force attack: A hash-based rfid mutual authentication protocol using a secret value," *Computer Communications*, vol. 34, no. 3, pp. 391–397, 2011.

[17] K. Bussi, D. Dey, M. Kumar, and B. Dass, "Neeva: A lightweight hash function," 2016.

[18] S.-S. Yeo and S. K. Kim, "Scalable and flexible privacy protection scheme for rfid systems," in *Security and Privacy in Ad-hoc and Sensor Networks.* Springer, 2005, pp. 153–163.

[19] S. Yu, K. Ren, and W. Lou, "A privacy-preserving lightweight authentication protocol for low-cost rfid tags," in *Military Communications Conference, 2007. MILCOM 2007. IEEE.* IEEE, 2007, pp. 1–7.

[20] M. Burmester and B. De Medeiros, "Rfid security: attacks, countermeasures and challenges," in *The 5th RFID Academic Convocation, The RFID Journal Conference*, 2007.

[21] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," in *Security in pervasive computing.* Springer, 2004, pp. 201–212.

[22] A. Juels, "Rfid security and privacy: A research survey," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 381–394, 2006.

[23] Y. Tian, G. Chen, and J. Li, "A new ultralightweight rfid authentication protocol with permutation," *Communications Letters, IEEE*, vol. 16, no. 5, pp. 702–705, 2012.

[24] E. G. Ahmed, E. Shaaban, and M. Hashem, "Lightweight mutual authentication protocol for low cost rfid tags," *arXiv preprint arXiv:1005.4499*, 2010.

[25] H.-Y. Chien, "Sasi: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity," *Dependable and Secure Computing, IEEE Transactions on*, vol. 4, no. 4, pp. 337–340, 2007.

[26] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estévez-Tapiador, and A. Ribagorda, "Lmap: A real lightweight mutual authentication protocol for low-cost rfid tags," in *Workshop on RFID security*, 2006, pp. 12–14.

[27] W. Shao-hui, H. Zhijie, L. Sujuan, and C. Dan-wei, "Security analysis of rapp an rfid authentication protocol based on permutation," *College of computer, Nanjing University of Posts and Telecommunications, Nanjing*, vol. 210046, 2012.

[28] T. Li and R. Deng, "Vulnerability analysis of emap-an efficient rfid mutual authentication protocol," in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on.* IEEE, 2007, pp. 238–245.

[29] R. C. Phan, "Cryptanalysis of a new ultralightweight rfid authentication protocol-sasi," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 4, p. 316, 2009.

[30] T. Li and G. Wang, "Security analysis of two ultra-lightweight rfid authentication protocols," in *New approaches for security, privacy and trust in complex environments.* Springer, 2007, pp. 109–120.

[31] 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.

[32] D. Ranasinghe, D. Engels, and P. Cole, "Security and privacy: Modest proposals for low-cost rfid systems," in *Auto-ID Labs Research Workshop, Zurich, Switzerland*, 2004.

[33] J. Huang, X. Li, C. Xing, W. Wang, K. Hua, and S. Guo, "Dtd: A novel double-track approach to clone detection for rfid-enabled supply chains."

[34] M. Lehtonen, F. Michahelles, and E. Fleisch, "How to detect cloned tags in a reliable way from incomplete rfid traces," in *RFID, 2009 IEEE International Conference on.* IEEE, 2009, pp. 257–264.

[35] M. Lehtonen, D. Ostojic, A. Ilic, and F. Michahelles, "Securing rfid systems by detecting tag cloning," in *Pervasive Computing.* Springer, 2009, pp. 291–308.

[36] D. Zanetti, L. Fellmann, and S. Capkun, "Privacy-preserving clone detection for rfid-enabled supply chains," in *RFID, 2010 IEEE International Conference on.* IEEE, 2010, pp. 37–44.

[37] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight mutual authentication and ownership transfer for rfid systems," in *INFOCOM, 2010 Proceedings IEEE.* IEEE, 2010, pp. 1–5.

[38] K. Mandal, X. Fan, and G. Gong, "Warbler: A lightweight pseudorandom number generator for epc c1 gen2 passive rfid tags," *Int. J. RFID Secur. Cryptogr*, vol. 2, pp. 82–91, 2013.

[39] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Lamed—a prng for epc class-1 generation-2 rfid specification," *Computer Standards & Interfaces*, vol. 31, no. 1, pp. 88–97, 2009.