Tailing RFID Tags for Clone Detection

Davide Zanetti
ETH Zurich, Switzerland
zanettid@inf.ethz.ch

Srdjan Capkun
ETH Zurich, Switzerland
capkuns@inf.ethz.ch

Ari Juels
RSA, The Security Division of EMC
ajuels@rsa.com

Abstract

RFID (Radio-Frequency IDentification) is a key emerging technology for supply-chain monitoring and detection of counterfeit and grey-market goods. The most prevalent RFID tags are, however, simply "wireless barcodes," themselves vulnerable to cloning and counterfeiting. While continuous monitoring can, in principle, detect cloning attacks, real-world supply chains often contain significant blind zones *where tag readings are unavailable, allowing attackers to inject counterfeit goods with cloned tags into supply chains undetectably.*

This paper introduces tailing*, a novel approach, both simple and practical, for detecting cloned RFID tags in supply chains. With tailing, RFID readers write random values to tags as they pass through a supply chain, creating in each tag a* tail *composed of random values. The tails of legitimate tags and cloned ones diverge over time, making cloning detectable by a centralized detector even across blind zones.*

We show that tailing works with existing barcode-type tags (e.g., EPC tags). The centralized detector is non-interactive, and requires no modification of existing supply-chain data flows. We characterize the cloning-detection efficacy of tailing analytically and through supply-chain simulations, showing that tailing presents high detection rates and low false positive rates, as well as rate tradeoffs outperforming those of previous schemes.

1 Introduction

Radio-Frequency IDentification (RFID) tags are inexpensive wireless microchips used to identify physical objects. RFID tags are present in passports, drivers' licenses, clothing, payment cards, and on shipping cases.

A major driver of the deployment of RFID systems is to prevent/detect *counterfeiting*, the introduction of fake goods into a supply chain. By affixing RFID tags directly to goods or the cases in which they are transported, supply-chain partners can automatically track goods in transit, facilitating detection of counterfeits. Counterfeit detection is of vital importance to many industries, such as the pharmaceutical industry, in which counterfeit goods have caused not only large profit losses, but also patient deaths [25]. Detection of counterfeit items in RFID-enabled supply chains may seem straightforward, since RFID tags typically emit unique identifiers: A supply-chain partner can in principle confirm an object's authenticity by checking its serial number against a shipping manifest or with a directory service spanning supply-chain partners, e.g., EPCIS [10]. However, two major challenges to RFID counterfeit detection remain: *Cloning attacks* and *fragmentary supply-chain visibility*.

Cloning attacks. Due to cost and power constraints, most RFID tags used in supply chains, known as EPC (Electronic Product Code) tags [11], have only "barcode"-like functionality. They emit raw data, with no authentication; their full data contents may then be easily extracted and copied into a special-purpose clone device or another tag [20]. Even tags with cryptography generally offer minimally effective tamper resistance or side-channel protections [12, 28].

Fragmentary supply-chain visibility. RFID tags are generally passive devices that transmit data only to nearby RFID readers. Their range is often limited to just tens of meters, and potentially further reduced by factors such as tag orientation, tag placement, and nearby materials (e.g., metal, water). Additionally, in supply chains, large populations of tags are often scanned in a short time (like a pallet of tagged goods passing through an RFID-enabled gate), causing read failures. Finally, some commercial partners cannot share supply-chain information or do not do so for fear of disclosing sensitive business intelligence. Entire segments of a supply chain may be opaque to participating entities. Thus, real-world supply chains often have large "blind zones," in which RFID tags scans do not happen or are not reported.

These two challenges undermine the effectiveness of unique identifiers in clone detection. Authentic identifiers do not ensure authentic tags or goods. Even natural detection strategies like looking for multiple, simultaneous appearances of the same tag identifier have limited effect. Blind zones can mask evidence of cloned tags or create inconsistencies in observed tag paths that lead to false alarms.

In this paper, we introduce a new approach to clone detection in RFID-enabled supply chains that we call *tailing*. Tailing consists of RFID readers writing random symbols into tags, creating in each tag a *tail* of values that evolves over time. Writing multiple symbols into the tags gradually randomizes the tails; this preserves symbol discrepancies over time, propagating them through blind zones. While passing through the supply chain, clones and authentic tags thus diverge in appearance, rendering clones more easily detectable. Tailing does not rely on any pre-defined (in)correct information based on supply-chain structure or product flow, which would make it sensitive to shifts in supply-chain dynamics. It relies instead on purpose-built evidence in the form of tails.

We analyze tailing both analytically and through simulations. To this end, we introduce an adversarial model suitable for the study of RFID clone-detection, which captures a broad space of adversarial capabilities such as reader compromise, and factors in chain configuration and visibility. To the best of our knowledge, it is the first supply-chain model in which adversaries can compromise readers.

We first explore tailing analytically by evaluating the probability of successful clone injection, i.e., undetected passage of a clone through the supply chain. We show that tailing diminishes the success probability of even strong adversaries that compromise many readers. We then evaluate the impact of blind zones and reader errors, showing the effectiveness of tailing in the face of the resulting *fragmentary visibility*.

Secondly, we holistically evaluate the security of tailing by simulating its use in various supply-chain scenarios with different degrees of visibility, chain structures, product flows, and adversaries. These simulations show that tailing achieves high clone-detection rates and low false-alarm rates even when visibility is highly fragmentary, as in real-world RFID-enabled supply chains. We further compare our mechanism against existing approaches [24, 36]; we find that tailing outperforms them in terms of true positive rate / false negative rate tradeoffs.

Finally, we show that tailing meets the resource requirements of ordinary, barcode-type RFID, such as EPC tags (e.g., tailing requires only 8 bits of tag memory). In addition, we show that tailing should scale with acceptable overhead in real-world supply chains.

The rest of the paper is organized as follows. In Section 2 we describe the considered supply-chain scenario and our system and adversarial models. We introduce tailing in Section 3 and provide an overview of our main results in Section 4. We present the results of analytic, simulation-based, and performance evaluations in Sections 5, 6, and 7 respectively. We compare tailing to existing work in Section 8, while Section 9 gives an overview of related work. We conclude the paper in Section 10.

2 Problem Statement

We now describe the considered RFID-enabled supply-chain scenario and present our system and adversarial models, which are depicted in Figure 1.

2.1 RFID-enabled Supply Chains

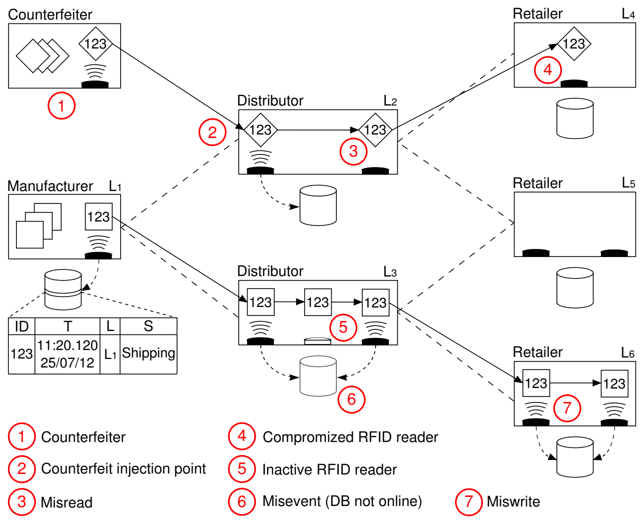
RFID-enabled supply chains are traditional supply chains enhanced such that each product (or pallet of products) is equipped with an RFID tag containing a unique identifier (ID). Supply-chain partners, like manufacturers, distributors, and retailers deploy RFID systems to create, store, and share observations of the tags/products circulating in the supply chain. An RFID system is typically composed of a front end, which includes RFID tags and readers, and a back end, which includes databases and service-oriented platforms like discovery and tracking services.

A product and its tag are considered to be a single, inseparable element. Tag hardware is constrained: Tags have limited memory and only basic functional capabilities. We assume no (cryptographic) authentication between tags and readers, as is the case with the EPC C1G2 standard [11] – a *de facto* RFID standard for supply chains. We also assume that the tag ID is not writable, but the tag memory can be read and (re)written by any nearby reader.

On each tag observation by a reader, an event is created and stored in a local database. An event encapsulates information about the process S (e.g., receiving, stocking, or shipping), occurrence time T, and location L in which a product/tag ID is sighted. Two special events are created when tags enter the supply chain (an *into-the-chain* event, created at the manufacturer when tags and IDs are assigned to products) and when tags leave the chain (*out-of-the-chain* event, created at the retailer). Each supply-chain partner is equipped with multiple RFID readers and local databases. Third party services may be authorized to access, aggregate, and analyze events from partners' local databases, typically to optimize business processes.

Our approach assumes failures in the front end during tag-reader communication, but that back-end failures are negligible. We also consider that some partners may not share data. We call it a *misevent* when an event is not reported (shared), a *misread* when a tag passes unnoticed (i.e., when no events are created), and a *miswrite* when a tag write operation fails (possibly corrupting memory). We do not consider broken or damaged tags and we assume that phantom reads are negligible and that multiple reads of the same tag are filtered out during data collection.

As for the supply-chain structure, we assume that partners know only their direct business partners and may continuously join and leave the chain. In terms of product flow, we consider recalls as well as misdeliveries.



ID	T	L	S
123	11:20.120 25/07/12	L_1	Shipping

Figure 1. An RFID-enabled supply chain affected by blind zones and reader failures (misevents, misreads, miswrites, inactive readers), as well as by counterfeiter's actions (clone injection, reader compromise), while a genuine product ($ID = 123$) and its clone are circulating in it.

2.2 System and Adversarial Models

The goal of the adversary is to inject counterfeit goods into the supply chain without detection by a centralized *detector*. This detector has a global view of the tags/goods in the supply chain: It collects and correlates events from the local databases of supply-chain partners. The adversary seeks to *hide* the presence of its counterfeits from the detector. We assume that all genuine products in the supply chain carry RFID tags. Therefore, a counterfeit product will pass as genuine only if it is equipped with a tag. Moreover, that tag must bear a valid and unique ID associated with a genuine product.

The visibility of the detector into the supply chain is limited to a subset of readers that participate in clone detection, and is affected by misevents, misreads, and miswrites. The detector does not rely on any pre-defined information about the supply-chain structure (e.g., partner relationships and locations) and product flow (e.g., transportation times).

The adversary controls a subset of readers. Such control models several forms of adversarial intrusion into the supply chain, including collusion with supply-chain partners, corruption of reader hardware/software, bribery of employees, and so forth. When the adversary compromises readers participating in the clone detection, it controls the channel to the detector and can dictate if and what data the latter receives. The adversary can inject new products into the supply chain with RFID tags bearing data of its choice and, additionally, knows valid identifiers for all tags in the supply chain at any given time. We call a *clone* a counterfeit product that carries a valid ID. The adversary may perform any of the following actions at the compromised readers:

- *Emulation:* The adversary may simulate the presence of an RFID tag, with data of its choice.
- *Blocking:* The adversary may prevent a compromised reader from scanning selected nearby RFID tags.
- *Tampering:* The adversary may alter the data contents of passing tags, but not tags' IDs, which are read-only.

The adversary also knows the paths followed by products in the supply chain. (Paths tend to be dictated by easy-to-ascertain commercial relationships, and are often fairly stable over time.) Thus it has the following capabilities:

- *Injection point selection:* The adversary can select counterfeit injection points.
- *Knowledge of genuine path:* The adversary knows the paths of genuine products.
- *Knowledge of counterfeit path:* The adversary knows the paths of counterfeit products.

The adversary is, however, restricted in three key regards:

- *No access to detector database:* The adversary cannot read or modify data gathered by the detector (but can *add* data to the database via emulation).
- *No product flow (path) modification:* The adversary cannot modify the paths followed by genuine goods in the supply chain (but can specify the paths taken by counterfeit products).
- *No product flow (lead time) modification:* The adversary cannot modify the product lead times (i.e., the times that products spend in the different steps of the supply chain like warehousing or transportation). The adversary does, though, learn the relative timestamps of the events for both genuine and counterfeit products. (For example, the adversary knows that reader i scans a genuine product before reader j scans the corresponding counterfeit.)

Finally, we assume that a counterfeit appearing before the genuine product enters, or after it leaves the supply chain is easily detected by verifying the corresponding into-the-chain and out-of-the-chain events. We refer to such a detection mechanism as *whitelist-based* detection.

3 Tailing for Clone Detection

Tailing relies on the creation and verification of traces of collected tag events. It requires collaboration between supply-chain partners and a service-oriented platform, the detector, and draws information from across the entire RFID system. It involves four different steps: (i) Tail modification, (ii) event collection, (iii) rule verification, and (iv) clone detection.

3.1 Tail Modification

As previously mentioned, a tag's tail consists of a sequence of random symbols that evolve over time. Tail *modification* is performed individually by each RFID reader participating in clone detection. It requires no interaction with the detector or other supply-chain partners (or other readers, for that matter). The operation of tail modification is an extension of the tag observation operation as detailed in Section 2.1, and affects both the tag memory and the observation event stored in the local database.

To modify a tag's tail, a reader refers to a stored value (in the tag memory) that we call the *tail pointer*, which points to the most recently modified tail position. It then writes a piece of random data (some random bits), a *symbol*, to the next available position in the tag memory, and increments and writes the new tail pointer value. The associated event created and stored in a local database includes, in addition to the ID, T, L, and S attributes, the tag tail TT and the tail pointer TP. It also includes a tailing *flag* TF, which indicates if the event is usable by the detector (i.e., that the reader is participating in tailing and the event appears to be valid). More precisely, to modify a tail, a reader: (i) Reads the tag ID, tail, and tail pointer from tag memory, (ii) updates the tail pointer (unitary increment, with wraparound) and stores it in the tag memory, (iii) picks a random symbol and inserts it in the tail, i.e., writes it to the next available position indicated by the pointer, and (iv) creates an event containing attributes $(ID, T, L, S, TT, TP, TF)$ and adds it to its local database.

Naturally, partners must agree on or standardize system parameters for use by the detector. A reader can at any time signal non-participation by marking its events as unusable using the tailing flag (TF).

3.2 Rule Verification and Clone Detection

Upon request by a supply-chain partner, the detector collects all of the events (with valid TFs) related to a specific tag ID to build what we call a tag *trace*. It validates this trace, looking for evidence of cloning, against a set of rules that we now describe.

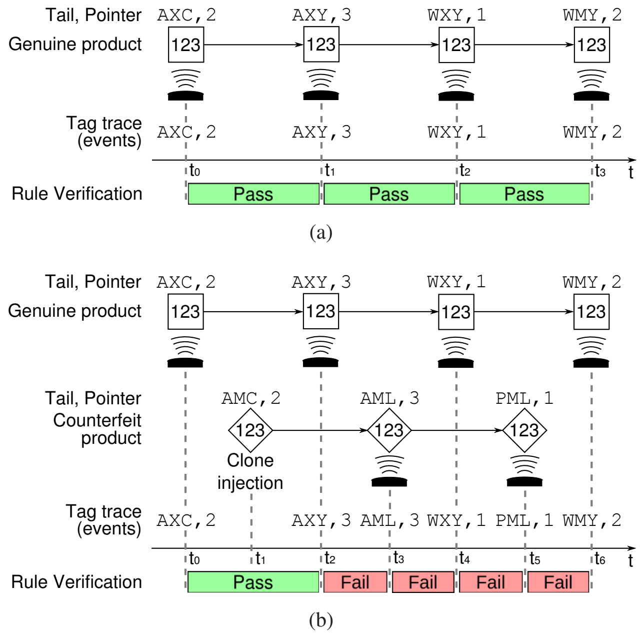


Figure 2. Rule verification on (a) the tag tail and pointer of a genuine product and (b) the tails and pointers of a genuine and a counterfeit product together. Values are given after a reader updates the tail and pointer. In this example, the tail is composed of 3 symbols. Each symbol is an 8-bit value (exemplified with ASCII characters).

Intuitively, a pair of events should reflect the results of valid tag modification operations. In particular, a pair composed of two time-consecutive events e_i and e_{i+1}, having tail and tail pointer equal to TT_i and TP_i, and to TT_{i+1} and TP_{i+1}, respectively, is considered correct if and only if: (i) The symbols in the tail TT_{i+1} are the same of those in TT_i (except in position TP_{i+1}), and (ii) the tail pointer TP_{i+1} presents a unitary increment with respect to TP_i. Formally, these two rules are:

$$\begin{cases} TT_{i+1}[n] = TT_i[n] & \forall n \setminus TP_{i+1} \\ TP_{i+1} - TP_i = 1 \pmod{t}, \end{cases} \tag{1}$$

where n ranges from 1 to tail size t (in symbols) and $TT[n]$ indicates the n-th symbol in the tail TT.

Figure 2(a) shows a (genuine) tagged product and its tag tail and tail pointer when it circulates in an RFID-enabled supply chain with tailing enabled (the tail and pointer values are shown after a reader's update). All the pairs of time-consecutive events are successfully verified, i.e., meet the two above rules. In contrast, Figure 2(b) shows two products with the *same tag ID*, i.e, a genuine and a counterfeit

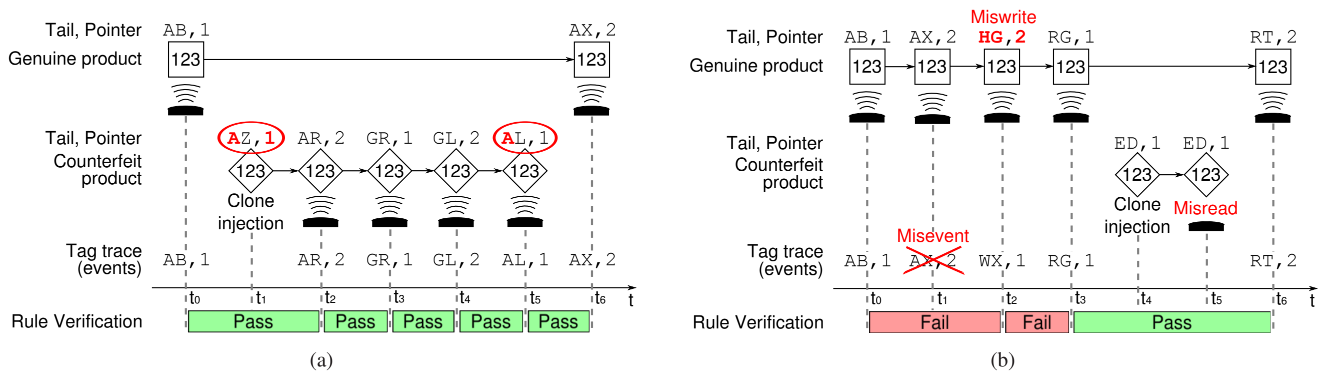


Figure 3. Tailing rule verification under (a) false event consistency and (b) weak visibility (misevents, miswrites leading to memory corruption, and misreads). To avoid clone evidence, in (a) the counterfeit tail and pointer at times t_1 and t_5 have to be *correctly* set.

product. Due to tail and pointer inconsistencies, some pairs fail the rule verification and show *evidence of cloning*.

3.3 Main Challenges

There are two main obstacles to successful detection via tailing: *False event consistency*, i.e., suppression or loss of clone evidence, and *weak visibility*, i.e., event streams that are fragmentary or error-prone.

False event consistency. Continuously updating tails with random values makes it difficult for the adversary to guess tail contents: Even if the adversary knows a tail value at time t_i, at time t_{i+j}, there are j new symbols to guess. Additionally, updating the tail and incrementing the pointer in an ordered sequence builds a relationship between two time-consecutive events: Any extra adversarial event between two time-consecutive events will break this relationship, resulting in clone evidence.

An adversary can nonetheless, with some probability, *cancel out* clone evidence. This is clearly possible when the adversary controls all the readers through which a counterfeit product (or the genuine one) passes. Even if the adversary has no control or partial control of readers, though, a chance remains that no clone evidence appears. Figure 3(a) illustrates, given a genuine and a counterfeit product, how this can happen. If the counterfeit tail and pointer are, by chance, consistent with the genuine tail and pointer at the injection time t_1 and at time t_5, no clone evidence would appear between events at times t_0 and t_2, and between events at times t_5 and t_6. By controlling only a subset of the readers, an adversary may increase the probability of such detection failure. For example, by controlling the reader that operates the genuine product at time t_0, the adversary would learn the genuine product tail and pointer and could ensure consistency at time t_1. For the adversary's clone now to pass undetected, the adversary would *only* need the counterfeit tail to be correctly set by chance at time t_5. Obviously, many factors (e.g., tail and symbol sizes) impact the probability of such undesired chance events.

Weak visibility. Blind zones and reader errors create fragmentary and erroneous visibility in the supply chain, masking evidence of clone tags or creating inconsistencies in the observed tag paths that lead to false alarms. The causes of fragmentary and erroneous visibility are misevents, misreads, and miswrites. A misevent occurs when a relevant event is not available to the detector. Misevents do not obstruct the tailing operation, i.e., the tag memory is correctly modified, but they result in lost events. As shown in Figure 3(b), this loss creates inconsistencies in tag traces as two reported time-consecutive events appear as non-consecutive, potentially raising a false alarm. Misevents may also mask clone evidence, as may misreads. A misread occurs when a tag passes unnoticed through a reader, so that no tail update or event creation results. As shown in Figure 3(b), misreads can lead to missed clones. A miswrite occurs when a tag write operation fails. Miswrites include cases when tags (i) reply with a write failure message (tag memory is not modified); alternatively, they may not report the result of the write operation and may have: (ii) Not modified, (iii) correctly modified, or (iv) incorrectly modified (corrupted) the tag memory. When a reader reports a miswrite by flagging an event as unusable using TF, case (iii) is comparable to a misevent, while cases (i) and (ii) are comparable to a misread. As shown in Figure 3(b), case (iv) creates inconsistencies that potentially raise a false alarm.

4 Overview of Main Results

We evaluate our tailing mechanism in terms of security and robustness through both an analytic (Section 5) and a simulation-based (Section 6) study. We also evaluate its performance in terms of required storage, computational effort, communication costs, scalability, and tag processing speed (Section 7). Finally, we compare it against previous work [24, 36] in a simulation-based study (Section 8).

In our analytic evaluation, we abstract the supply chain and the product flow into two independent event sequences associated, respectively, with a genuine product and its clone. For each of the possible combinations of these sequences and a given adversary, we compute the probability that a combination does not present any clone evidence. By summing probabilities across combinations, we quantify the adversary's success probability in injecting clones, i.e., the probability that injected clones pass unnoticed by tailing detection.

In the simulation-based evaluation, we model a supply chain by its structure (partners and their relationships), product paths, lead times, and RFID system (readers and failures). We deploy a custom-built RFID-enabled supply-chain simulator to generate a flow of products from the manufacturers (both genuine and counterfeit) to the retailers, and to populate the detector's database with tag events recorded by each partner's reader during the simulation. We then compute the clone-detection and false-alarm rates when products leave the chain at the retailers.

Analytic evaluation. We show that our tailing mechanism drastically limits the adversary's success probability, even when the majority of the readers in the clone path have been compromised. For example, blocking 2, 3, or 4 out of 5 readers in the clone path leads to an adversary's success probability of 0.81, 3.6, and 12.5% respectively. We also show that the optimal symbol size is 1 bit, while a tail larger than 5 symbols does not provide any significant advantage over adversaries that compromise readers. This means that our mechanism requires a limited tag memory space as small as 8 bits (5 bits for the tail, 3 bits for the tail pointer). Additionally, we compute an upper bound on the adversary's success probability by considering an adversary that can select the optimal strategy (in terms of readers to compromise and actions to perform) for each injected clone. Although such a strong and arguably non-realistic adversary presents high(er) success probabilities, e.g., 15, 52, and 90% when compromising 2, 3, or 4 readers respectively (5 on the clone path), our tailing mechanism is still able to detect a fraction of the injected clones.

Our mechanism relies on purpose-built, artificial information that is independent from the supply-chain structure and the product flow. That is, it does not require any predefined (in)correct information such as "Product X has to go through locations L_x, L_y, and L_z at times t_x, t_y, and t_z." Thus our mechanism is unaffected by extraordinary flow deviations and changes in the supply-chain structure, due, for example, to product recalls, misdeliveries, and partners joining and leaving the chain. Although blind zones and reader failures (i.e., misevents, misreads, and miswrites) negatively impact our mechanism, we show that it mitigates the negative effects of misevents and miswrites (false alarms) with no (significant) increase in the adversary's success probability. Misevents are mitigated by hypothesizing missing events between two reported time-consecutive events that present tail and pointer inconsistencies. If there exists at least one missing event that would resolve these inconsistencies, the ostensible clone evidence is discarded. Such flexibility is possible because our tailing mechanism updates the tail and increments the pointer in an ordered sequence, allowing for trace reconstruction. Miswrites are mitigated by specifically reporting write failures through an additional event attribute. A reader that does not receive a correct write response from a tag sets this attribute in the corresponding event. The detector will then ignore the inconsistencies resulting from that event and succeeding one. We find that misreads create clone misses that may significantly increase the adversary's success probability. We show, however, that even for a high number of misreads, our mechanism is still able to detect a portion of the injected clones. For example, an adversary that compromises 4 out of 5 readers in the clone path presents a success probability of 56% even when half of all events are subject to misreads.

Simulation-based evaluation. We show that our mechanism presents a high detection rate for a relatively low false-alarm rate, as well as a relatively high detection rate for a false-alarm rate of 0. For example, in a scenario where an adversary injects clones with no reader compromise and more than 60% of all traces contain at least one inconsistency due to misevents, misreads, or miswrites, we observe detection rates of 93% and 80% for false-alarm rates of 0.95% and 0% respectively. This result holds for different supply-chain structures and clone injection rates. We also show that our mechanism presents good detection and false-alarm rates in scenarios where a large majority of the tag traces (85%) present inconsistencies: We observe detection rates of 86% and 64% for false-alarm rates of 2.8% and 0.01% respectively. We observe that our mechanism is affected by the length of clone paths: The longer the path, the higher the number of instances of clone evidence. Even within scenarios where clone paths are short (e.g., when the adversary injects at the retailers), our detection mechanism still provides a high detection rate (>85%) for a relatively low false-alarm rate (0.95%). We also observe that although an adversary compromising readers can significantly reduce the detection rate, only controlling the (quasi) totality of all clone paths leads to no detection at all. For example, in a simulated scenario with a 15-partner supply chain, we obtain detection rates of 74% and 6% for an adversary controlling 3 and 12 of the 15 partners, respectively.

Performance evaluation. Our mechanism's resource costs are independent of the overall number of tags in the system. Instead, its required storage capacity, computational / accessing effort, and communication costs are linearly dependent on the number n of events in the traces under evaluation; thus the mechanism's resource costs also scale linearly with n. The tag processing speed (i.e., speed at which tag IDs can be read) is affected by the tailing operation between a tag and reader, which includes (tail and pointer) read and write operations on tag memory. Despite the limited tag-memory requirement of our mechanism, it reduces the nominal tag processing speed in an EPC C1G2-compliant implementation from 24.4/1838 tags/s (lower/upper bound) to 9.6/44.5 tags/s. We argue, however, that tailing can be performed by a few readers in a supply-chain facility (e.g., only upon product receipt and shipping); thus it is a rare operation with little overall processing overhead. Consequently, we believe that tailing promises to be highly scalable in real-world supply-chain environments.

Comparison. We compare our tailing mechanism against the mechanisms proposed by Lehtonen et al. [24] and Zanetti et al. [36]. Our exploration indicates that both mechanisms present limitations that make our tailing mechanism the most suitable solution for scenarios in which inconsistent information may undermine clone detection. This superiority holds not only by comparison with the mentioned mechanisms, but also for their robust variants that we propose and explore in our evaluation. The mechanism proposed by Lehtonen et al. suffers mainly from false alarms due to misevents and miswrites. Its robust variant mitigates miswrite effects, but is not effective against misevents. The mechanism proposed by Zanetti et al. suffers mainly from false alarms due to misevents and misreads. Although its robust variant mitigates both misevent and misread effects, no improvement in its detection/false-alarm tradeoff results. In a scenario where an adversary injects clones with no reader compromise and more than 60% of all traces contain at least one inconsistency, we observe detection/false-alarm rates of 89%/32%, 94%/16%, and 93%/0.95% for the Lehtonen et al., the Zanetti et al., and our tailing mechanisms, respectively. With their false-alarm rates set to 0%, their respective detection rates drop to 37%, 34%, and 80%.

All three mechanisms present the same (asymptotic) required database storage capacity, computational effort, communication costs, and scalability. However, despite its poor detection/false-alarm tradeoff, the mechanism proposed by Zanetti et al. has the benefit of not relying on tags' rewritable memory, and thus avoids a mechanism-dependent degradation of the nominal tag processing speed (max. 1838 tags/s). Although the mechanism proposed by Lehtonen et al. presents severely limited tag processing speeds due to its online execution (max. 4.5 tags/s), an offline variant offers the same performance as tailing.

5 Analytic Evaluation

We now evaluate the success probability of an adversary in injecting a counterfeit product into a supply chain given the use of tailing (Section 5.2). We also evaluate the robustness of tailing against misreads, miswrites, and misevents (Section 5.3).

5.1 Definitions

We model the supply chain $\mathcal{S}$ as an acyclic, directed graph $H = (V, K)$, in which nodes V represent readers and edges K represent supply-chain paths between readers. Products enter the graph / supply chain at a source node and traverse edges along a path π until they reach the sinks.

The detector $\mathcal{D}$ has limited visibility into the supply chain, corresponding to a subset of readers $V_D \subseteq V$. We define $G = (g_0, ..., g_m)$ as the sequence of detection-relevant events associated with a genuine product following a path π_G with start node $v_{G,0}$ and sink $v_{G,m}$. V_G indicates the nodes in π_G ($V_G \subseteq V_D$). For a clone product, we define $C = (c_0, ..., c_n)$, π_C (with start node $v_{C,0}$ and sink $v_{C,n}$), and V_C ($V_C \subseteq V_D$) analogously.

The detector performs event collection at some time t_D, triggered by a product reaching its sink. We define an event trace GC as a time-sorted combination of the sequences G and C at the detection time t_D. We consider g_0 to be the into-the-chain event for the genuine product; therefore, the first event of a combination GC is g_0. We define $T_{GC}(G, C) = (GC_0, ..., GC_j)$ as the set of all possible combinations of the sequences G and C. The set size $|T_{GC}|$ is equal to $\binom{|G|+|C|+1}{|C|+1} - 1$.[1]

Each reader may fail with probability p_{mw} in a tag write operation and with probability p_{mr} in a tag read operation. The detector receives an event in G or C with probability p_{me}. Thus we define $\widehat{GC}$ as the *actual* event trace received by the detector at detection time t_D.

Trace evidence. We define P_{pass} as the probability that a trace $\widehat{GC}$ contains no clone evidence according to the detector $\mathcal{D}$ and given adversary $\mathcal{A}$. $E = (\epsilon_0, ..., \epsilon_i)$ denotes clone evidence instances observed under the rule set of Equation 1 based on a tailing operation with tag tail size t (in symbols), tail symbol size s (in bits), and tail pointer size $p = log_2(t)$. The size m of the tag memory dedicated to the detection mechanism is equal to $t \cdot s + \lceil p \rceil$ bits. We specify P_{pass} as:

$$P_{pass} = \prod_{i=0}^{|E|-1} P_{\epsilon_i} = \left(\frac{1}{2^{(t-1)\cdot s+p}}\right)^{a_0} \cdot \prod_{i=1}^{|E|-1} \left(\frac{1}{2^{(t-1)\cdot s}}\right)^{a_i} \cdot b_i.$$

[1]Combination with repetitions $\binom{n+k-1}{k}$ for $n = |G|+1, k = |C|+1$.

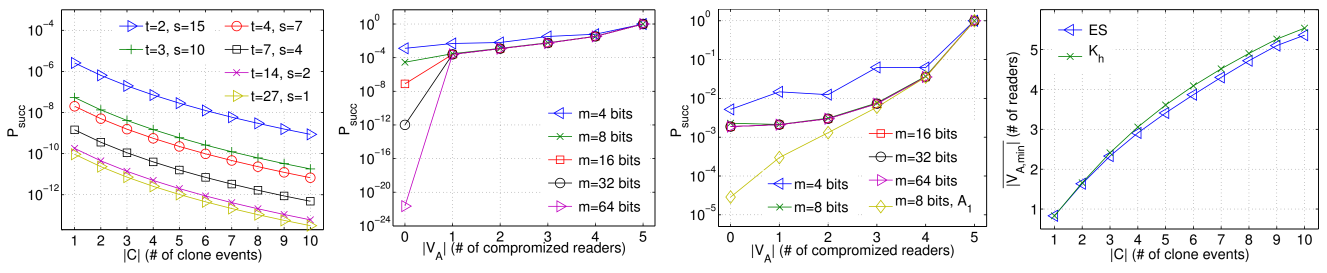


Figure 4. Study of clone events and compromised nodes for 1bit-tailing ($|G| = 10\ events$). Graphs show: (a) Success probability P_{succ} for adversary $\mathcal{A}_0$ as a function of number $|C|$ of clone events for various values of tail size t and tail symbol size s ($m = 32\ bits$); (b) P_{succ} for adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ as a function of compromised nodes $|V_A|$ for various tag memory sizes m ($s = 1\ bit$, $|C| = 5$, and $|V_A| = 0$ for $\mathcal{A}_0$); (c) Same as graph (b), but for $\mathcal{A}_2$; (d) Mean number of readers that $\mathcal{A}_3$ must compromise to suppress evidence fully in a set $T_{GC}(G, C)$, as a function of $|C|$, for strategies $\mathcal{K}_h$ and ES.

Here, P_{ϵ_i} is the probability that clone evidence ϵ_i passes unnoticed (does not appear); $a_i = 0$ when adversary $\mathcal{A}$ cancels out evidence ϵ_i (otherwise $a_i = 1$); $b_i = 0$ when the number n of (clone or genuine) events between evidence ϵ_{i-1} and ϵ_i is not a multiple of t (otherwise $b_i = 1$). The first term ($i = 0$) captures the probability that a clone is injected with correct tail and pointer values (w.r.t. the genuine product). The second term ($i > 0$) captures the probability that consistent genuine and clone tails (and pointers) remain consistent after n tailing operations on one of the two tails (i.e., lack of clone evidence persists).

Adversary's success probability. We define P_{succ} as the probability that a clone injected by adversary $\mathcal{A}$ into the supply chain at node $v_{C,0} \in V$ is not observed by the detector $\mathcal{D}$. This success probability sums over all possible combinations of the sequences G and C. It is defined as:

$$P_{succ} = \sum_{i=0}^{|T_{GC}|-1} P_{GC}(GC_i) \cdot P_{pass}(\widehat{GC}_i, \mathcal{A}),$$

where $P_{GC}(GC_i)$ is the probability associated with combination GC_i under probability distribution P_{GC}.

Adversary. Under the model of Section 2.2, we define a hierarchy of adversaries, ordered by increasing capability:

$\mathcal{A}_0$: The adversary injects clones into the supply chain at some selected node $v_{C,0}$, but does not compromise readers or influence product paths.

$\mathcal{A}_1$: The adversary injects clones at some selected node $v_{C,0}$ and compromises a set of readers V_A (chosen randomly) on the clone path π_C ($V_A \subseteq V_C \subset V_D$). Since the adversary has no knowledge of the genuine path π_G, it only prevents compromised readers from scanning clones (i.e., blocks them).

$\mathcal{A}_2$: Along with $\mathcal{A}_1$'s capabilities, the adversary can eavesdrop on genuine path π_G so as to inject clones with correct memory content, i.e., $P_{\epsilon_0} = 1$ for all GC_i.

$\mathcal{A}_3$: The adversary knows relative event timestamps and also knows genuine paths and forces its clones to follow them. After injecting a clone with correct memory contents at some selected node $v_{C,0}$, it can compromise any reader in V_D to cancel out clone evidence in a sequence GC_i. The adversary can abuse readers to prevent tag scanning (blocking), inject fake events into local databases (emulation), and modify tag memory contents (tampering). Readers are selected and misused according to a strategy $\mathcal{K}$ detailed in the next section.

5.2 Security Evaluation

In this section we evaluate the security of tailing in terms of success probability P_{succ} for the above-described adversaries. We let P_{GC} be uniformly distributed.

Figure 4(a) shows the success probability P_{succ} for adversary $\mathcal{A}_0$ as a function of the number of clone events $|C|$ and over different combinations of tail size t and symbol size s (dedicated tag memory size $m = 32$ bits and number of genuine events $|G| = 10$). The best performing combination of (t, s) is $t = 27$ symbols and $s = 1$ bit. Intuitively, with 1-bit symbols, the adversary has to guess all bits in the tail but one, resulting in maximal uncertainty. For the rest of our study, we only consider 1-bit symbols and call this *1bit-tailing*.

The impact of the number of nodes $|V_A|$ that an adversary compromises and of different memory sizes m is shown in Figures 4(b) and 4(c) for adversaries $\mathcal{A}_0$-$\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. (Here, $|G| = 10$, $|C| = 5$; for $\mathcal{A}_0$, $|V_A| = 0$.) For $\mathcal{A}_1$ and $\mathcal{A}_2$, compromising nodes dramatically raises the adversary's success probability, even eliminating the benefits of a larger tag memory ($m > 8$ bits). For $\mathcal{A}_0$, though, the larger m, the lower P_{succ}. By injecting clones with correct memory contents, $\mathcal{A}_2$ achieves the highest P_{succ} values.

As adversary $\mathcal{A}_3$ can compromise readers, it defines a lower bound on the number of compromised readers needed to cancel out clone evidence in a tag trace. In fact, it may: (i) Block all events between evidence instances ϵ_{i-1} and ϵ_i or (ii) between ϵ_i and ϵ_{i+1}, (iii) inject fake events between events e_j and e_{j+1} that lead to ϵ_i, or (iv) tamper with the memory of the tag that generates e_{j+1}. Actions (i) and (ii) aim to remove events that create inconsistencies, while actions (iii) and (iv) aim to create consistent event transition. Different strategies can be deployed to suppress evidence in a sequence GC; Figure 4(d) shows, for varying numbers of clone events, the mean minimum number of readers that adversary $\mathcal{A}_3$ has to compromise in order to suppress evidence in every trace GC_i. Two choices of strategy $\mathcal{K}$ are depicted. Strategy ES represents the optimal solution (in terms of sequences of permitted $\mathcal{A}_3$ actions) computed by exhaustive search individually for each trace GC_i. Strategy $\mathcal{K}_h$ is a heuristic derived from study of ES action sequences. $\mathcal{K}_h$ suppresses clone evidence ϵ_i by injecting fake events between events e_j and e_{j+1}; if there is just one event between ϵ_i and ϵ_{i+1}, $\mathcal{K}_h$ instead blocks e_{j+1}. Interestingly, this simple strategy yields results comparable to the optimal solution ES. Both strategies highlight the power of $\mathcal{A}_3$ to successfully suppress evidence with relatively few readers, e.g., 5.5 readers on average for $|C| = 10$ events.

Table 1 summarizes the percentage of the combinations in T_{GC} that lead to $P_{pass} = 1$ (i.e., that present no clone evidence) for the four adversary types ($|G| = 10$ events and $|C| = 5$ events; valid for any m). For the weakest adversary, $\mathcal{A}_0$, all combinations in T_{GC} present clone evidence. The highest P_{pass}, equal to $2^{-[(t-1)\cdot s+p]}$, is obtained only for those $|G|+|C|$ combinations (15 combinations, or 0.19% of the total) with a single and uninterrupted sequence of clone events at the end. For $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, some combinations have $P_{pass} = 1$ (thus their higher P_{succ} values). For $\mathcal{A}_1$, being able to block $|V_A|$ nodes ($V_A \subset V_C$) allows full suppression of clone evidence in $\binom{|G|+|V_A|}{|V_A|} - 1$ combinations out of the $|T_{GC}|$ possible ones, corresponding to the alignment of $|V_A|$ clone events with compromised readers. For example, for $|V_A| = 3$, $\mathcal{A}_1$ is able to suppress evidence fully in all of the 285 combinations (3.6% of the total) that contain only events (c_0), (c_0, c_1), or (c_0, c_1, c_2). For $\mathcal{A}_2$, having $\mathcal{A}_1$'s capabilities plus injecting clones with correct tails allows successful doctoring of a set of $\binom{|G|+|V_A|}{|V_A|} - 1 + |G| + |C| - |V_A|$ combinations, corresponding to the union of sets leading to the highest P_{pass} for $\mathcal{A}_0$ and $\mathcal{A}_1$. We note that adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ are most likely to achieve success in cases where they randomly compromise the *leading* nodes in the path π_C, i.e., from $v_{C,0}$ to $v_{C,|V_A|-1}$. In fact, for $m \geq 8$ bits, values in Table 1 correspond to upper bounds on P_{succ} for $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_3$ is more powerful in that it can compromise readers adaptively; thus it achieves the highest P_{succ} values, applicable as upper bounds for all adversary types.

Table 1. Percentage of combinations in the set $T_{GC}(G, C)$ that lead to $P_{pass} = 1$ under adversaries $\mathcal{A}_0$ to $\mathcal{A}_3$. $|G| = 10$ *events* and $|C| = 5$ *events*.

	$\lvert V_A\rvert$ (# of compromised readers)					
	0	1	2	3	4	5
$\mathcal{A}_0$	0%	-	-	-	-	-
$\mathcal{A}_1$	0%	0.12%	0.81%	3.6%	12.5%	100%
$\mathcal{A}_2$	0.19%	0.3%	0.97%	3.7%	12.6%	100%
$\mathcal{A}_3$	0.19%	2.2%	14.8%	52%	89.9%	100%

5.3 Robustness Evaluation

In this section we evaluate the robustness of 1bit-tailing against misevents, misreads, and miswrites.

Misevents cause inconsistencies in (genuine) tag traces, which then raise false alarms.[2] Figure 5(a) (curve $em = 0$) shows the ratio of false alarms as a function of the misevent probability p_{me} for a trace composed of 10 genuine events. To mitigate the effect of misevents, the set of rules in Equation 1 can be relaxed to accept a number em of missing events between two (reported) time-consecutive events. In this relaxation, a pair of events e_i and e_{i+1} is considered correct for which: (i) The symbols in the tail TT_{i+1} are the same as those in TT_i, except for the symbols rewritten between positions TP_i (exclusive) and TP_{i+1} (inclusive), and (ii) the difference between the two tail pointers is at most em plus one (the one corresponding to the pointer increment). Formally, Equation 1 is then extended to:

$$\begin{cases} TT_{i+1}[n] = TT_i[n] \begin{array}{ll} \forall n \backslash \;\; (TP_i, TP_{i+1}] & \text{if } \Delta_{TP} > 0 \\ \forall n \in \;(TP_{i+1}, TP_i] & \text{if } \Delta_{TP} < 0 \end{array} \\ \\ \Delta_{TP} \pmod t \leq me + 1, \end{cases}$$

[2]Misevents may also cause clone misses. We evaluate this effect when considering misreads.

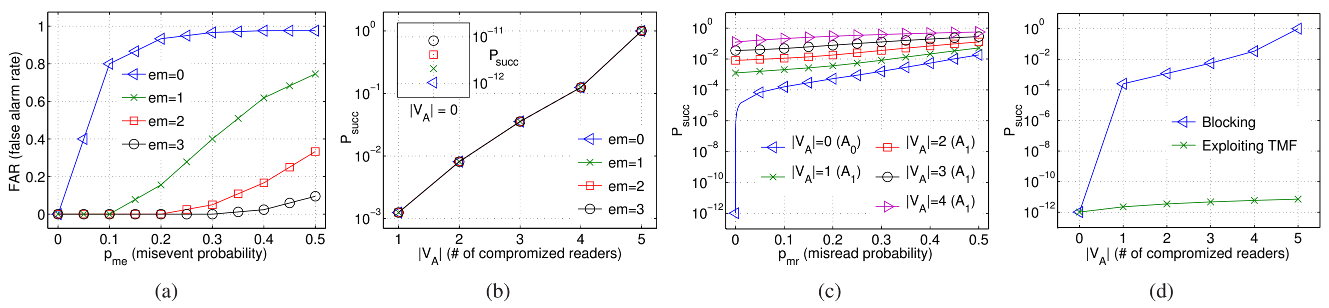


Figure 5. Study of misevents, misreads, and miswrites in 1bit-tailing. Graphs are: (a) False-alarm rate as a function of misevent probability p_{me} for a trace composed of 10 genuine events and for various values of em, the number of allowed missing events between consecutive trace events; (b) Success probability P_{succ} for $\mathcal{A}_0$ and $\mathcal{A}_1$ as a function of compromised nodes $|V_A|$ for various values of em (with $\mathcal{A}_0$, $|V_A| = 0$); (c) P_{succ} for $\mathcal{A}_0$ and $\mathcal{A}_1$ as a function of misread probability p_{mr} for various numbers $|V_A|$ of compromised nodes; (d) P_{succ} for $\mathcal{A}_1$ that either blocks clone events (*Blocking*) or exploits the miswrite flag (*Exploiting TMF*) to mark clone events as affected by miswrites. For all graphs, $|G| = 10\ events$, $|C| = 5\ events$, and $m = 32\ bits$.

where n ranges from 1 to tail size t, $TT[n]$ indicates the n-th symbol in the tail TT, and Δ_{TP} is equal to $(TP_{i+1} - TP_i)$.[3]

Figure 5(a) shows the strong impact of different em values on the false-alarm rate. Allowing missing events could have the side effect of increasing the adversary's success probability. However, as shown in Figure 5(b), adversary $\mathcal{A}_0$ ($|V_A| = 0$) only marginally benefits from $em > 0$. The adversarial benefit of compromised nodes is overwhelming compared to that of allowing missing events; adversaries $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ do not benefit significantly from $em > 0$.

Misreads may mask clone events and cause the detector to miss clones. As shown in Figure 5(c), even a small misread probability p_{mr} significantly increases $\mathcal{A}_0$'s success probability; in fact, due to missing clone events, a few traces do not present any clone event (i.e., $P_{pass} = 1$) even for $\mathcal{A}_0$. However, for small p_{mr} (< 0.1), $\mathcal{A}_0$'s P_{succ} is still relatively low ($< 10^{-4}$); it becomes significant ($> 10^{-2}$) only for large p_{mr} (> 0.5). Although the success probability for adversary $\mathcal{A}_1$ is primarily a function of the number of compromised readers, misreads also contribute in increasing P_{succ}. This contribution becomes significant for relatively large p_{mr}. (The same holds for $\mathcal{A}_2$ and $\mathcal{A}_3$).

Miswrites include cases when tags (i) reply with a write failure message (no memory modification); they also include cases when tags do not acknowledge a failed write operation and have (ii) not modified, (iii) correctly modified, or (iv) incorrectly modified (corrupted) the tag memory. If miswrites are reported through the tailing flag *TF*, cases (i) and (ii) are comparable to misreads, and case (iii) to a misevent. To mitigate the effect of memory corruption, i.e., case (iv), we extend tag events with a *miswrite flag* (*TMF*). A reader not receiving any write operation result from a tag will set both the tailing and the miswrite flags in an event e_i; the detector will then ignore a clone evidence between events e_i and e_{i+1}. Although the miswrite flag could be misused by an adversary to mark clone events as miswrites, as Figure 5(d) shows (for $\mathcal{A}_1$), this provides no advantage in terms of P_{succ} over blocking clone events. In fact, misusing the miswrite flag only cancels out clone evidence resulting from a clone event followed by a genuine event, but not vice versa; blocking can cancel both. These observations hold also for $\mathcal{A}_2$. For $\mathcal{A}_3$, misusing the miswrite flag is equivalent to injecting fake events; no additional advantage in terms of P_{succ} is achieved.[4]

We refer to our mechanism as *basic 1bit-tailing* (*TAIL*) when $em = 0$ and the miswrite flag *TMF* is not deployed, and *robust 1bit-tailing* (*R-TAIL*) when $em = 3$ and the flag *TMF* is deployed.

6 Simulation-based Evaluation

We further evaluate our tailing mechanism through a custom-built, RFID-enabled supply-chain simulator. This allows us to explore more realistic and complex scenarios than those in Section 5.

[3]For $\Delta_{TP} = 0$, the tail has been completely rewritten, which makes the result of the rule verification not reliable. It may be also possible that the tail is completely rewritten even for $\Delta_{TP} \neq 0$. In order to avoid such situation, the tail size t should be larger than $\lceil |G| \cdot p_{me} \rceil + 1$.

[4]Injecting fake events may suspiciously increase the number of events in a trace, though, while misuse of the miswrite flag does not.

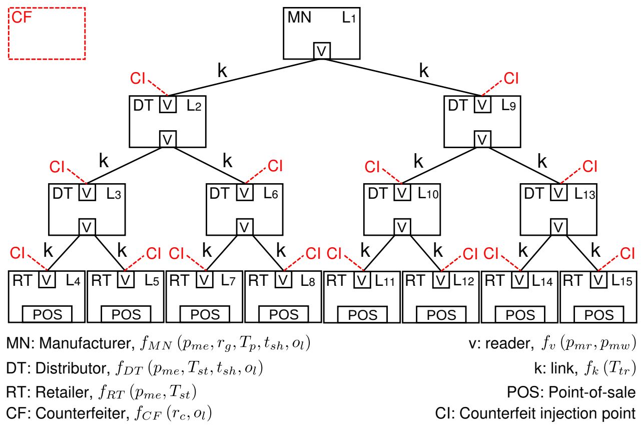

CF
MN L1
V
CI
k
DT V L2
DT V L9
DT V L3
DT V L6
DT V L10
DT V L13
RT V L4
RT V L5
RT V L7
RT V L8
RT V L11
RT V L12
RT V L14
RT V L15
POS
MN: Manufacturer, $f_{MN}\,(p_{me}, r_g, T_p, t_{sh}, o_l)$
DT: Distributor, $f_{DT}\,(p_{me}, T_{st}, t_{sh}, o_l)$
RT: Retailer, $f_{RT}\,(p_{me}, T_{st})$
CF: Counterfeiter, $f_{CF}\,(r_c, o_l)$
v: reader, $f_v\,(p_{mr}, p_{mw})$
k: link, $f_k\,(T_{tr})$
POS: Point-of-sale
CI: Counterfeit injection point

Figure 6. 4-level binary-tree supply chain with one manufacturer (MN), two 2nd-level distributors (DT), four 3rd-level distributors (DT), and 8 retailers (RT). A counterfeiter (CF) injects clone products at different points (CI) in the chain. L, v, k, and POS stand for location, reader, link, and point-of-sale respectively. The relation between each element and the simulation parameters summarized in Table 2 is also shown.

Table 2. Simulation parameters for the considered baseline scenario. $\mathcal{N}(\mu, \sigma)$ represents a normal distribution.

	Parameter	Value
p_{mr}	Misread probability	$\mathcal{N}(5\%, 1\%)$
p_{mw}	Miswrite probability	$\mathcal{N}(5\%, 1\%)$
p_{me}	Misevent probability	$\mathcal{N}(5\%, 1\%)$
r_g	Genuine production rate	1000 products/day
r_c	Counterfeit production rate	10 products/day
T_p	Production time	2 months
t_{sh}	Shipping time	1/day at 8AM
T_{st}	Stocking time	$\mathcal{N}(3, 0.5)$ days
T_{tr}	Transport time	$\mathcal{N}(1, 0.25)$ days
o_l	Output load (demand)	Uniformly distributed
S	Supply-chain structure	4-level binary tree
m	Tag memory size	16 bits
(t, s)	Tail and symbol sizes	$(12, 1)$ bits
Adversary		$\mathcal{A}_0$
Counterfeit injection point		Random at any partner

6.1 Description and Baseline Scenario

Our simulator generates a flow of genuine products in a supply chain from the product manufacturer to one or several distributors, and finally retailers. It also generates a flow of counterfeit products from injection points (potentially any partner in the chain) to retailers. The product flow is defined by the supply-chain structure, product demand, and lead times (stocking time within a partner, shipping time from a partner, and transport time between two partners). Each partner has RFID readers that record events and may perform tailing. Clone detection occurs when products leave the chain, i.e., at retailers' points-of-sale.

As a baseline scenario for our study, we define a supply chain of 15 partners (and locations) distributed in a 4-level binary tree (Figure 6). Participating readers are those associated with receiving and shipping operations. We consider EPC C1G2 tags, which operate on 16-bit data blocks (Section 7); tail and symbol sizes of 12 symbols and 1 bit respectively allow use of just one such block. The counterfeiter is adversary $\mathcal{A}_0$ (Section 5.1). It can simply inject counterfeit products with valid IDs. In our baseline scenario, it does so at a randomly selected partner, excluding the manufacturer, as soon as a new and valid ID is obtained. (In practice, a counterfeiter injects clones by posing as a legal seller and obtaining valid IDs right after the genuine products enter the chain.) Table 2 summarizes the parameters for this baseline scenario.

Detection depends on the number $|E|$ of instances of suspected clone evidence in a given trace, i.e., pairs of time-consecutive events that fail the rule verification stage. Cloning is suspected if $|E| \geq DT$, for a parameterized *detection threshold* DT. For each scenario, 10 runs are executed, each over 2 months of production.[5] A trace contains all events for genuine and clone products with a given ID until one such product reaches a point-of-sale. (The second product to reach a point-of-sale eventually triggers an alarm under the basic whitelist-based detection.)

6.2 Experimental Results

We let TPR and FPR respectively denote the *true positive rate* and *false positive rate* for a given setting.

Figure 7(a) shows the superior performance of robust 1bit-tailing (*R-TAIL*) over basic 1bit-tailing (*TAIL*).[6] (As de-

[5]For each system element (e.g., a reader, Figure 6), the value of its parameters that refer to some probability distributions (e.g., the misread probability) is drawn from the given probability distribution at each run.

[6]Small variance can be observed when considering different simulation runs. This is valid for all the simulation-based results in this paper.

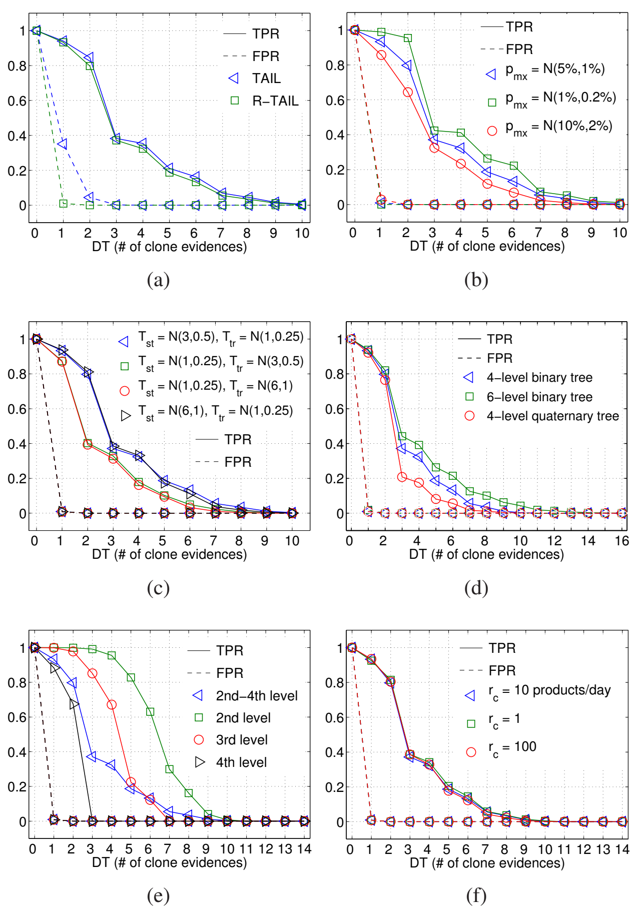


Figure 7. Detection rate (TPR) and false-alarm rate (FPR) as a function of detection threshold DT for *R-TAIL* in (a) the baseline scenario (Section 6.1) and while varying: (b) Misread, miswrite, and misevent probabilities ($p_{mx} = p_{mr} = p_{mw} = p_{me}$); (c) Stocking and transport times (T_{st} and T_{tr}, in days); (d) Supply-chain structure; (e) Counterfeit injection point (supply-chain level); (f) Counterfeit production rate (r_c).

fined in Section 5.3.) *TAIL* is more sensitive to inconsistencies generated by miswrites and misevents than *R-TAIL*; it occasionally detects a cloned ID when direct clone evidence is effaced by misreads. But this sensitivity yields substantially higher false-alarm rates than *R-TAIL* with nearly TPRs. The best results occur with a small DT: With $DT = 1$, *R-TAIL* detects 93% of clones with a false-alarm rate of 0.95%; with $DT = 2$, the false-alarm rate drops to 0% for a detection rate of 80%. ($DT > 2$ merely lowers the TPR.) Given these results, *all experiments that follow use* R-TAIL *only*.

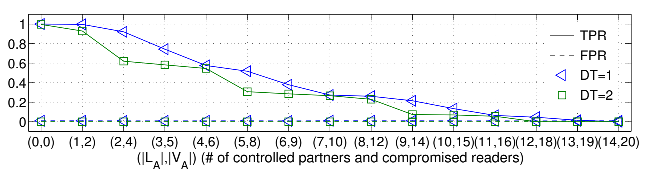


Figure 8. Detection rate (TPR) and false-alarm rate (FPR) as a function of detection threshold DT for *R-TAIL* in the baseline scenario (Section 6.1), but considering adversary $\mathcal{A}_2$ that controls $|L_A|$ locations (partners) and compromises $|V_A|$ readers.

Figure 7(b) shows the impact of different misread, miswrite, and misevent probabilities on detection and false-alarm rates ($p_{mr} = p_{mw} = p_{me} = p_{mx} = \mathcal{N}(\mu, \sigma)$, i.e., each probability is normally distributed with μ and σ). As the rate of misreads and misevents rises, of course, counterfeit products are more likely to go unnoticed, decreasing the detection rate. Even under *highly adverse* conditions, though, e.g., $p_{mx} = \mathcal{N}(10\%, 2\%)$ (which translates into 85% of traces presenting at least one inconsistency), *R-TAIL* detects 86% of all clones with a false-alarm rate of 2.8% ($DT = 1$); with a false-alarm rate of 0.01%, the detection rate is 64% ($DT = 2$).

The impact on the detection and false-alarm rates of the stocking and transport times (T_{st} and T_{tr}) is shown in Figure 7(c). Varying T_{st} and T_{tr} does not impact the false-alarm rate. $T_{tr} > T_{st}$, though, decreases the detection rate: Under this condition, a counterfeit product injected at a retailer can emerge (exits the chain) before the genuine product reaches a 2nd-level partner.

As shown in Figure 7(d), the supply-chain structure has a limited impact on both rates for $DT = 1$, and a larger impact for $DT > 1$. The main determinant of the detection rate is the (average) length of the paths traversed by products: Structures with longer paths (e.g., a 6-level binary tree) generate more clone evidence than those with shorter paths (e.g., a 4-level binary tree).

Similarly, counterfeit goods are harder to detect when injected toward the end of the supply chain, and thus traversing fewer partners, as shown in Figure 7(e). Finally, detection and false-alarm rates are invariant to the rate of injection/production of counterfeit goods (Figure 7(f)).

Figure 8 shows the impact on detection and false-alarm rates of the stronger adversary $\mathcal{A}_2$, as defined above in Section 5.1. $\mathcal{A}_2$ controls a subset L_A of supply-chain locations (partners) and the full set of readers within these locations. $\mathcal{A}_2$ injects clones at the 2nd-level distributors (Figure 6) and compromises locations so as to control the maximum number of complete manufacturer-retailer paths. (E.g., for

$|L_A| = 4$, $\mathcal{A}_2$ controls locations L_2 to L_5.) Curves here represent the TPR and FPR for the detection thresholds leading to the highest detection rate ($FPR = 0.95\%$, $DT = 1$) and the lowest false-alarm rate ($FPR = 0$, $DT = 2$) as a function of the number $|L_A|$ of supply-chain partners under the control of the adversary, as well as of the number $|V_A|$ of compromised readers. An aggressive adversary can significantly impact detection rates: Compromise of 50% of all readers reduces the detection rate to 27%.

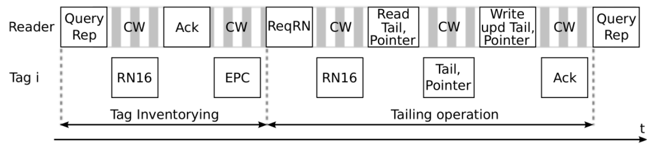


Figure 9. EPC C1G2-compliant tailing operation and tag inventorying for one tag. CW stands for continuous waveform.

7 Performance Evaluation

In this section we evaluate the resource costs of tailing in terms of storage, computation, and communication costs, as well as its performance in terms of tag processing speed. We assume use of EPC C1G2 RFID tags [11].

An RFID reader executing *R-TAIL* must read the entire tag tail and tail pointer from tag memory and write an updated pointer and a new symbol. A single EPC C1G2 write operation operates on a data block of 16 bits, while a single read operation operates on up to 128 bits. In a minimal configuration (8 bits for tail and pointer), a tailing operation requires a single read and a single write operation. Figure 9 shows the corresponding communication sequence between a reader and a tag in an inventorying and tailing operation. The following performance results.

Storage capacity. Tailing consumes little memory, even by the standards of low-cost tags, e.g., 8 bits. A local database requires storage equal to the number of events generated by the reader(s), i.e., tailing creates no additional events. Each event, though, has to be extended with tail-related attributes *TT*, *TP*, *TF*, and *TMF*, resulting in a minimum 7% increase in the event size for an 8-bit tag dedicated memory.[7] Use of an EPC C1G2 tag's full minimal data block size, i.e., 16 bits, results in event size increasing by 10%.

Computation. Tags perform no computation, while readers perform only lightweight operations, e.g., pseudo-random or random number generation for bit updates. The detector needs to perform the rule evaluation on each pair of two time-consecutive events, a form of basic complex-event-processing (CEP) that imposes fairly little computational overhead.

Communication. Tailing requires readers to perform extra write operations (see below on tag processing speed), but carries no extra cost on back-end communication between local databases and the detector;[8] however, as for its storage overhead for events, messages have to additionally include the tail-related attributes.

[7]Assuming a basic event that comprises 28 bytes: (ID(12 bytes), T(8 bytes), L(4 bytes), P(4 bytes)). For one-byte storage granularity, an 8-bit dedicated memory requires 2 bytes of storage, and a 16-bit one, 3 bytes of storage.

[8]We assume that events are already collected by other service-oriented platforms to optimize business processes.

Tag processing speed. The EPC C1G2 standard requires a nominal tag processing speed for inventory operations (reads) of 24.4 to 1838 tags/s, depending on several parameters (e.g., the tag data rate, encoding scheme, and data modulation). As noted above, tailing requires writing into tag memory, currently a time-intensive operation: The EPC C1G2 standard allows a write time up to 20 ms for a single 16-bit memory block. With an 8-bit dedicated memory, writing a new tail bit and updating the tag pointer (4 bits in total) would require writing an entire memory block, resulting in a nominal tag processing speed of 9.6/44.5 tags/s. Commercial tags, however, outperform EPC specifications; e.g., the Impinj Monza 5 chip [1] supports a write speed of approx. 2.5 ms, potentially boosting processing speed to 11.5/200 tags/s. (Other tags, we believe, are similar.)

In summary, tailing carries little overhead. Its main cost is the slowdown on reader-to-tag communication caused by tag writes. However, tailing can be performed by a few readers in a supply-chain facility (e.g., upon product receipt and shipping as explored in Section 6), which makes it a rare operation. Consequently, we believe that tailing promises to be highly scalable in real-world supply-chain environments.

8 Comparison

Using our supply-chain simulator from Section 6.1, we now compare tailing against the clone detection schemes of Lehtonen et al. [24] and Zanetti et al. [36].

The Lehtonen et al. [24] scheme resembles tailing: It writes a fresh, *single* random value in tag memory at each tag observation. A tag event includes the old (read) value and the fresh (written) one. In contrast to tailing, this scheme relies on an *online* centralized entity that creates and stores tag events and verifies consistency between them. If a previously written value (stored in event e_{i-1}) is identical to the currently read one (event e_i), the tag and the centralized entity are said to be *synchronized*; otherwise a clone alert is raised. We refer to this scheme as *synchronization* (or *SYNCH*). The Zanetti et al. [36] scheme verifies the correctness of consecutive shipping and receiving operations using

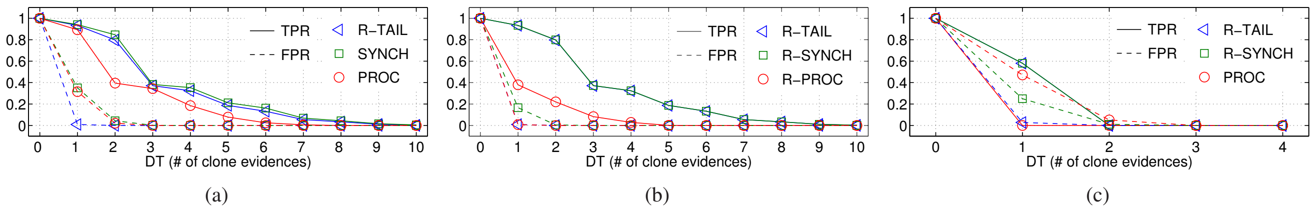


Figure 10. Detection rate (TPR) and false-alarm rate (FPR) as a function of detection threshold DT for robust 1bit-tailing (*R-TAIL*), synchronization (*SYNCH* and *R-SYNCH*), and process-based (*PROC* and *R-PROC*) schemes for: (a,b) The baseline scenario (Section 6.1); (c) A more challenging, *highly adverse* scenario with $p_{mr} = p_{mw} = p_{me} = \mathcal{N}(10\%, 2\%)$, $T_{st} = \mathcal{N}(1, 0.25)$ $days$, $T_{tr} = \mathcal{N}(6, 1)$ $days$, counterfeit products injected at the retailers, and a 4-level quaternary-tree chain structure.

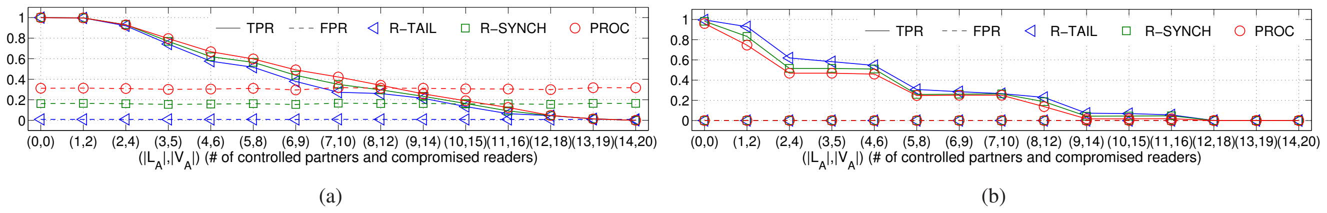


Figure 11. Detection rate (TPR) and false-alarm rate (FPR) as a function of detection threshold DT for robust 1bit-tailing (*R-TAIL*), robust synchronization (*R-SYNCH*), and process-based (*PROC*) schemes in the baseline scenario (Section 6.1), but considering adversary $\mathcal{A}_2$ that controls $|L_A|$ locations (partners) and compromises $|V_A|$ readers. Graphs show rates for (a) the highest detection rate ($DT = 1$) and (b) the lowest false-alarm rate ($FPR = 0$).

process and location information in tag events (attributes S and L). In particular, it verifies: (1) That a product shipped from location A (event e_i with $S = shipping$ and $L = A$) is received at some location B (e_{i+1} with $S = receiving$ and $L = B$), or (2) that a product received at location A (e_i with $S = receving$ and $L = A$) is then shipped from the same location (e_{i+1} with $S = shipping$ and $L = A$). If neither condition is met, an alarm is raised. Unlike tailing and *SYNCH*, this approach involves no modification of tag memory or attribute extension to tag events. Like tailing, though, tag events are stored in local databases and collected later for rule evaluation and clone detection. We refer to this scheme as *process-based* (or *PROC*).

As in Section 6, we consider EPC C1G2 tags operating on 16-bit data blocks. *R-TAIL* thus uses a tail and symbol size of 12 symbols and 1 bit respectively, while *SYNCH* uses a 16-bit synchronized value (nominally 32 bits in [24]).

8.1 Security and Robustness

Figure 10(a) shows clone-detection and false-alarm rates (TPR and FPR) across schemes (*R-TAIL*, *SYNCH*, and *PROC*), as a function of detection threshold DT in the baseline scenario (Section 6.1). *R-TAIL* and *SYNCH* achieve comparable TPRs, but both somewhat outperform *PROC* (93% vs. 89% for $DT = 1$). *R-TAIL*, however, achieves substantially lower FPR than *SYNCH* and *PROC*: 0.95% vs. 35% and 32% respectively ($DT = 1$).

Like *TAIL*, *SYNCH* is affected by misevents and miswrites, which lead to false alarms. In contrast, it lacks linkage between non-time-consecutive tag observations (as provided by the pointer in tailing), which implies no mitigation for misevents. We can introduce a miswrite flag to mitigate miswrites, however, yielding a refined scheme that we call *robust synchronization* (*R-SYNCH*). As in *R-TAIL*, the two-rule system of *PROC* can be relaxed to allow em missing events; for example, a pair of events e_i and e_{i+1} having, respectively, ($L = A, S = receiving$) and ($L = B, S = receiving$) will not raise an alarm if an event with ($L = A, S = shipping$) is assumed to be missing between them. This would mitigate both misevents and misreads leading to false alarms. We call this new scheme *robust process-based* (*R-PROC*). Figure 10(b) shows the effect of robustness in these new schemes. It yields a lower FPR in the synchronization scheme of Lehtonen et al. (16% vs. 35% for $DT = 1$), although still higher than for tailing (16% vs. 0.95% for $DT = 1$); the impact on TPR

Table 3. Comparison between detection schemes. FN and FP stand for False Negatives and False Positives respectively. Tag speeds are nominal (see Section 7 for discussion).

	Baseline scenario (TPR, FPR)	(Section 6.1) (TPR, FPR)	Misreads impact	Misevents impact	Miswrites impact	Event-size overhead	Tag memory [bits]	Tag speed [tags/s]
PROC [36]	(89%, 32%)	(34%, 0%)	FN, FP	FN, FP	-	0%	-	24.4–1838
R-PROC	(38%, 0.95%)	(22%, 0%)	FN	FN	-	0%	-	24.4–1838
SYNCH [24]	(94%, 35%)	(38%, 0%)	FN	FN, FP	FP	14%	16	3.3–4.5
R-SYNCH (offline)	(93%, 16%)	(37%, 0%)	FN	FN, FP	FN	14%	16	9.6–44.5
R-TAIL	(93%, 0.95%)	(80%, 0%)	FN	FN	FN	10%	16	9.6–44.5

is negligible. In contrast, for the process-based scheme of Zanetti et al., robustness lowers *both* FPR *and* TPR, such that *PROC* presents better tradeoffs than *R-PROC*; e.g., for $FPR = 0\%$, *R-PROC* achieves $TPR = 22\%$ compared with $TPR = 34\%$ for *PROC*. Thus, we henceforth use *R-SYNCH* and *PROC* in our experimental comparisons.

Figure 10(c) compares the three schemes *R-TAIL*, *R-SYNCH*, and *PROC* in a *highly adverse* scenario, with the simulation settings yielding pessimal performance in Section 6: High misread, miswrite, and misevent probabilities ($p_{mr} = p_{mw} = p_{me} = \mathcal{N}(10\%, 2\%)$), long transport time ($T_{tr} = \mathcal{N}(6, 1)$ days), short stocking time ($T_{st} = \mathcal{N}(1, 0.25)$ days), counterfeit products injected at retailers, and a 4-level quaternary tree. Here, tailing achieves the best performance, with a better FPR than *R-SYNCH* (2.8% vs. 25% for $DT = 1$) and globally better TPR and FPR than *PROC*.

Figure 11(a) explores a stronger adversary $\mathcal{A}_2$ (as in Section 6.2 above). It shows the TPR and FPR as a function of the number $|L_A|$ of adversarially controlled supply-chain locations (partners) and the number $|V_A|$ of compromised readers; here, we choose $DT = 1$, which yields the highest detection rates. Tailing has the lowest TPR (e.g., 27%, vs. 42% for *PROC* and 35% for *R-SYNCH* for $(|L_A| = 7, |V_A| = 10)$—but uniformly with substantially lower FPR (0.95% vs. 17% and 31%). A more balanced comparison is achieved with a normalized FPR, for which tailing achieves the *highest* TPRs. Figure 11(b) compares the three schemes for FPR = 0%.

8.2 Cost Comparison

All three schemes impose asymptotic storage, computation, and communication costs linear in the number of trace events n evaluated by the detector. Storage costs are a function of the number of tag observations, as is communication between readers and the back-end detector, while computation is dominated by the execution of rule evaluation sequentially over events.

Synchronization requires extending each event with old and fresh synchronization values, 32 bits in total, while tailing requires 3 bytes (16 bits for the tail and pointer, 8 bits for the flags), leading to event size increases of 14% and 10% respectively (see Section 7).

Unlike tailing and synchronization, *PROC* requires no tag writes, and thus has no impact on tag processing speeds. In an EPC-compliant implementation, it can reach the nominal values of 24.4/1838 tags/s. Tailing and synchronization require tag writes. Synchronization further requires online interaction with the centralized entity, imposing network latency of about 200 ms [24] and limiting processing speed to a mere 4.5 tags/s. It is possible, though, to modify *R-SYNCH* (to resemble *R-TAIL*) so that readers generate random values and detection occurs offline. Both schemes then have a nominal top tag processing speed of about 44.5 tags/s. (Higher performance is likely in practice: See Section 7.)

8.3 Summary

Table 3 summarizes our key results and presents the distinct vulnerabilities of the various schemes to misreads, misevents, and miswrites, further illuminating their respective performance. *R-TAIL* outperforms other approaches, with a high detection rate for a relatively low false-alarm rate, and a distinctly high detection rate for FPR = 0%.

9 Related Work

Anti-counterfeiting solutions based on track-and-trace data within RFID-enabled supply chains were initially discussed by Kuh et al. [18] and Staake et al. [33], who highlighted the negative impact on counterfeit detection of incomplete traces when partners do not record or share tracking data. More recently, several solutions based on verifying the (in)correctness of event traces and tag behaviors have been proposed [3, 9, 17, 22–24, 27, 36]. These solutions rely on intrusion detection, classifying activity based on pre-defined models of suspicious patterns (misuse) and normal tag/product behaviors (anomaly). Mirowski et al. [27] apply statistical anomaly detection to identify RFID tag ownership changes indicative of theft or cloning of tags based on reader operations, tag and reader IDs, and event timestamps. Similarly, Lehtonen et al. [22, 23] explore anomaly and anomaly/misuse intrusion detection, accommodating incomplete traces caused by tag misreads and partners not sharing tag observations. Kerschbaum and Oertel [17] propose a pattern-matching approach to detect illicit transactions between supply-chain partners. Blass et al. [3] and Elkhiyaoui et al. [9] leverage tag memory to store verifiable tag paths (visited readers). While all of these mechanisms are suitable for low-cost (EPC C1G2) tags, they require training or deep knowledge of supply-chain structures and product flows, causing fragility in the face of supply-chain changes, product recalls, and product misdeliveries. Moreover, only Lehtonen et al. [22,23] consider incomplete (but still not faulty) traces.

Closest to our work are Lehtonen et al. [24] and Zanetti et al. [36]. These schemes do not rely on pre-defined information about chain structures and product flows, but have the limitations demonstrated in Section 8. Thus tailing proves the most effective for typical supply-chain scenarios in which inconsistent information (due to blind zones and reader failures) complicates clone detection. Additionally, other solutions do not consider adversarial abilities beyond injection of cloned tags (e.g., reader compromise).

Other approaches to detect/prevent tag cloning include physical-layer fingerprinting techniques, authentication based on low-cost primitives or cryptographic ones, and Physical Unclonable Functions (PUFs). As shown in [29, 35], EPC C1G2 tags can be fingerprinted with high accuracy over the air at the physical layer, without added tag hardware. Such fingerprinting is sensitive to environmental factors, though, limiting its use in supply-chain scenarios. Low-cost authentication protocols that exploit native EPC C1G2 computation have also been proposed [5,6,19], but follow-up work has identified significant weaknesses [26, 30, 31]. Use of native EPC C1G2 storage and access control is a complementary approach [16], but is vulnerable to eavesdropping attacks. PUFs [13] are low-complexity (hundreds of gates), purpose-built circuits that exploit manufacturing variations for authentication. Several low-cost PUF-based security solutions have been proposed [8, 21, 34], but recent attacks highlight the need for better understanding of tradeoffs among PUF circuit size, security level, and stability [32]. Several solutions leveraging symmetric- and public-key cryptography have been proposed for RFID tags [2]. High-security crypto primitives, though, are prohibitively expensive for low-cost tags today, requiring a few thousand gate-equivalents for symmetric-key primitives as in, e.g., [7], and more for public-key primitives, e.g., [14]. Additionally, most of these cryptographic solutions introduce non-trivial key-management challenges.

10 Conclusion

As the use of RFID as an anti-counterfeiting technology in supply chains grows, cloning attacks against tags, exacerbated by blind zones, are a pressing systemic vulnerability. We have shown that tailing is a simple and a practical countermeasure and is effective even across blind zones, where the centralized detector lacks visibility into tag emissions.

Tailing outperforms previous anti-cloning schemes. In a 4-level supply-chain simulation, for instance, we observe detection rates (93%) equal to the next best approach (synchronization), but with much better false positive rates (0.95% vs. 16%). In addition, tailing requires limited tag resources and no extra infrastructure resources and introduces minimal overhead on supply-chain processes.

We believe that tailing is a potent new tool meriting exploration in other settings. For instance, tailing may help detect cloning of post-supply-chain goods carried by consumers, such as luxury goods. (An interesting new RFID privacy challenge then arises: Abuse of tails as “cookies.”) Similarly, tailing might supplement cryptographic and other anti-cloning protections. RFID-enabled payment devices and travel documents, both shown vulnerable to cloning attacks [4, 15, 20], are attractive potential beneficiaries.

References

[1] `http://www.impinj.com/`.

[2] `http://www.avoine.net/`.

[3] E.-O. Blass, K. Elkhiyaoui, and R. Molva. Tracker: Security and privacy for RFID-based supply chains. In *NDSS*, 2011.

[4] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, and M. Szydlo. Security analysis of a cryptographically-enabled RFID device. In *USENIX Security Symposium*, 2005.

[5] C.-L. Chen and Y.-Y. Deng. Conformation of EPC Class 1 Generation 2 standards RFID system with mutual authentication and privacy protection. *Engineering Applications of Artificial Intelligence*, 22, 2009.

[6] H.-Y. Chien and C.-H. Chen. Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards. *Computer Standards & Interfaces*, 29, 2007.

[7] M. David. *Lightweight cryptography for passive RFID tags*. PhD thesis, Aalborg University, 2011.

[8] 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 *IEEE RFID*, 2008.

[9] K. Elkhiyaoui, E.-O. Blass, and R. Molva. CHECKER: On-site checking in RFID-based supply chains. In *ACM WiSec*, 2012.

[10] EPCglobal. EPCIS Standard v. 1.0.1. Standard, 2007.

[11] EPCglobal. UHF Class 1 Gen 2 standard v. 1.2.0. Standard, 2008.

[12] F. Gandino, B. Montrucchio, and M. Rebaudengo. Tampering in RFID: A survey on risks and defenses. *Mobile Networks and Applications*, 15(4), 2010.

[13] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *ACM CCS*, 2002.

[14] D. Hein, J. Wolkerstorfer, and N. Felber. ECC is ready for RFID - A proof in silicon. In *SAC*, 2008.

[15] T. S. Heydt-Benjamin, D. V. Bailey, K. Fu, A. Juels, and T. O'Hare. Vulnerabilities in first-generation RFID-enabled credit cards. In *FC*, 2007.

[16] A. Juels. Strengthing EPC tags against cloning. In *ACM WiSe*, 2005.

[17] F. Kerschbaum and N. Oertel. Privacy-preserving pattern matching for anomaly detection in RFID anti-counterfeiting. In *RFIDSec*, 2010.

[18] R. Koh, E. W. Schuster, I. Chackrabarti, and A. Bellman. Securing the pharmaceutical supply chain. White paper, Auto-ID Labs, MIT, 2003.

[19] D. M. Konidala, Z. Kim, and K. Kim. A simple and cost-effective RFID tag-reader mutual authentication scheme. In *RFIDSec*, 2007.

[20] K. Koscher, A. Juels, V. Brajkovic, and T. Kohno. EPC RFID tag security weaknesses and defenses: Passport cards, enhanced drivers licenses, and beyond. In *ACM CCS*, 2009.

[21] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan. Lightweight mutual authentication and ownership transfer for RFID systems. In *IEEE INFOCOM*, 2010.

[22] M. Lehtonen, F. Michahelles, and E. Fleisch. Probabilistic approach for location-based authentication. In *IWSSI*, 2007.

[23] M. Lehtonen, F. Michahelles, and E. Fleisch. How to detect cloned tags in a reliable way from incomplete RFID traces. In *IEEE RFID*, 2009.

[24] M. Lehtonen, D. Ostojic, A. Ilic, and F. Michahelles. Securing RFID systems by detecting tag cloning. In *Pervasive*, 2009.

[25] T. Mackey and B. Liang. The global counterfeit drug trade: Patient safety and public health risks. *Journal of Pharmaceutical Sciences*, 100(11), 2011.

[26] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti. A practical implementation attack on weak pseudorandom number generator designs for EPC Gen2 tags. *Wireless Personal Communications*, 59(1), 2011.

[27] L. Mirowski and J. Hartnett. Deckard: A system to detect change of RFID tag ownership. *IJCSNS*, 7(7), 2007.

[28] Y. Oren and A. Shamir. Remote password extraction from RFID tags. *IEEE Transaction on Computers*, 56(9), 2007.

[29] S. C. G. Periaswamy, D. R. Thompson, and J. Di. Fingerprinting RFID tags. *IEEE TDSC*, 8(6), 2011.

[30] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and J. C. A. van der Lubbe. Cryptanalysis of an EPC C1G2 standard compliant authentication protocol. *Engineering Applications of Artificial Intelligence*, 24(6), 2011.

[31] P. Peris-Lopez, T. Li, T.-L. Lim, J. C. Hernandez-Castro, and J. M. Estevez-Tapiador. Vulnerability analysis of a mutual authentication scheme under the EPC Class-1 Generation-2 standard. In *RFIDSec*, 2008.

[32] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *ACM CCS*, 2010.

[33] T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC network: The potential of RFID in anti-counterfeiting. In *ACM SAC*, 2005.

[34] P. Tuyls and L. Batina. RFID-tags for anti-counterfeiting. In *CT-RSA*, 2006.

[35] D. Zanetti, B. Danev, and S. Capkun. Physical-layer identification of UHF RFID tags. In *ACM Mobicom*, 2010.

[36] D. Zanetti, L. Fellmann, and S. Capkun. Privacy-preserving clone detection for RFID-enabled supply chains. In *IEEE RFID*, 2010.