Privacy-preserving Clone Detection for RFID-enabled Supply Chains

Davide Zanetti, Leo Fellmann, and Srdjan Čapkun
Dept. of Computer Science, ETHZ, Zurich, Switzerland
{zanettid, fellmannl, capkuns}@inf.ethz.ch

***Abstract*—Counterfeit products cause financial losses and represent a health risk. Within RFID-enabled supply chains, where products are equipped with RFID tags, clone detection mechanisms based on tag traces can help in detecting counterfeits. These mechanisms assume that supply chain partners share (private) information to run trace analysis, and may suffer from supply chain dynamics, tag misreads, product recalls, and misdeliveries. In this work, we present a novel, effective, privacy-preserving clone detection mechanism for RFID-enabled supply chains. Our mechanism does not rely on global knowledge of supply chain structures or products flow, it is robust to recalls and misdeliveries, and considers tag misreads while evaluating the presence of counterfeits. We propose privacy-preserving implementations of our mechanism that show better performance when compared to similar implementations based on existing secure multi-party computation frameworks.**

I. Introduction

Counterfeit products injected into legal supply chains affect both supply chain partners and final consumers, causing financial losses (estimated USD 200 billion of counterfeits internationally traded in 2005 [1]) and representing health risks (e.g., through counterfeit pharmaceuticals). Within Radio Frequency Identification (RFID)-enabled supply chains, products are equipped with RFID tags containing unique identifiers. Through an RFID infrastructure (e.g., the EPCglobal network [2]), supply chain partners can record, store, and share information associated with those identifiers, and use it to increase automation, speed-up processes, and enable new services like anti-counterfeiting [3] and real-time tracking [4]. In order to successfully inject counterfeits into legal RFID-enabled supply chains, those have to be equipped with tags containing valid identifiers; from the point of view of the RFID infrastructure, this makes counterfeits copies, i.e., *clones*, of genuine products.

Several mechanisms have been proposed for both preventing and detecting tag cloning. Mainly, those can be categorized into cryptographic tag-authentication solutions and trace-based solutions [5]. The former aim at making tag cloning harder through the use of cryptographic primitives and encryption schemes that provide reliable authentication and preserve secrets; this is the most common approach, but it requires additional hardware resources and key management strategies [6]. The latter aim at detecting clones by verifying tag behaviors against predefined rules (e.g., verifying the plausibility of a tag's visited locations); this does not require expensive tag resources and is therefore well suited for low-cost tags. It is, however, based on tag-related data distributed among the supply chain partners. These partners are reluctant to share the owned data due to concerns about the possible use of that data to infer sensitive information like strategic information (e.g., volumes), relationships, unfair behaviors or inefficiencies, and distribution channels [7]. These concerns can seriously compromise the data sharing, leading to situations in which none, or only few partners actually share their data. Therefore, to enable and thus benefit from optimizations and services based on distributed and private tag-related data, it is fundamental to guarantee the privacy of this data while providing these optimizations and services. Additionally, trace-based solutions may suffer from real-world supply chain phenomena like supply chain dynamics (i.e., partners continuously joining and leaving the chain and changing business relationships), tag misreads[1], and extraordinary events like product recalls and misdeliveries, which affect tag traces by introducing noise that may be interpreted by predefined rules as caused by clones.

In this paper, we address the aforementioned limitations to trace-based solutions and propose a novel, trace-based, privacy-preserving clone detection mechanism with a clone detection rate of 85.5% for 0.1% false alarm rate. Our mechanism verifies the correct sequence of tag observations related to transport processes, i.e., shipping and receiving. It does not rely on global knowledge of supply chain structures or products flow, which makes it robust to supply chain dynamics, recalls, and misdeliveries, and mitigates the effect of tag misreads by considering the misread probability when evaluating the presence of clones. We provide data privacy through ad-hoc cryptographic protocols for secure multi-party computation (SMC) based on threshold homomorphic encryption, oblivious transfer, garbled circuits, and mixnets. This enables supply chain partners to deploy our mechanism without disclosing information on neither their observations nor relationships. Although existing SMC frameworks (e.g., [9], [10]) could implement secure computations for any probabilistic polynomial-time function, they can be impractical in many relatively complex scenarios. Our privacy-preserving protocols are optimized for scenarios in which a relatively high number of partners (>5) securely evaluate the presence of clones. Our protocols guarantee security against semi-honest adversaries, do not rely on trusted third parties, and present time and bandwidth improvements up to 75% compared to implementations of our clone detection mechanism within existing SMC frameworks.

Our contributions are as follows: (i) we design a novel, effective, trace-based clone detection mechanism for RFID-enabled supply chains which considers tag misreads, does not

[1]Tag read rate is affected by tags' orientation, position, surrounding material, and density. For example, different tagged elements like empty and rice-filled boxes can lead to (non-optimized) read rates of 99% and 80% respectively [8].

rely on global knowledge of supply chain structures or products flow, and is robust to supply chain dynamics, recalls, and misdeliveries, (ii) we design and implement ad-hoc, privacy-preserving protocols for the proposed mechanism that enable clone detection and preserve partners data privacy, and (iii) we evaluate these protocols and show their improvements over similar implementations within existing SMC frameworks.

To the best of our knowledge, this work is the first to address data privacy concerns for trace-based clone detection solutions.

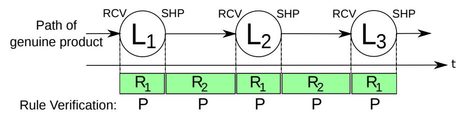
The rest of this paper is organized as follows. We introduce the considered supply chain scenario in Section II. We present and evaluate our clone detection mechanism in Sections III and IV respectively. In Section V, we introduce the necessary cryptographic toolkit. We present and evaluate our privacy-preserving protocols in Sections VI and VII respectively. In Section VIII, we give an overview of background and related work. We conclude the paper in Section IX.

II. Scenario: RFID-enabled Supply Chains

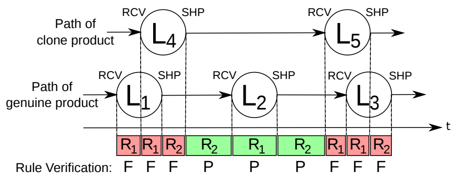
We consider an RFID-enabled supply chain where each product is equipped with an RFID tag containing a unique identifier (ID). A product and its tag are considered inseparable, and tags contain only IDs; no additional information is stored on the tags. Moreover, we assume no cryptographic-based authentication between tags and readers (compliant with the EPC C1G2 standard [11], which is a *de facto* standard for supply chain applications) and we do not consider broken or damaged tags.

Supply chain partners include manufacturers, wholesalers, and retailers. All partners are equipped with RFID readers and back-end infrastructure (time-synchronization between partners is assumed), and record and store events associated to products in local and private databases. Events are the direct consequence of tag observations (i.e., a reader that reads a tag); in local databases, an event is a tuple that encapsulates four attributes (ID, T, L, S), that correspond to the process (S, e.g., receiving, stocking, or shipping), relative occurrence time (T), and location (L) in which a product/tag (ID) has been involved. Two special events are created when tags enter the supply chain (i.e., at the manufacturer, when tags and IDs are assigned to products) and when tags leave the chain (i.e., at the retailer). Events are considered private/sensitive information and contain real and not intentionally or involuntarily corrupted/damaged/wrong information. Partners may publish event-related information (e.g., ID + partner's database address) to a discovery service (DS), which allows partners to retrieve information about who owns events for which tag. Data stored in partners and DS databases can be accessed through authentication and access control mechanisms (e.g., based on a PKI infrastructure). Partners only know their direct business partners and may continuously join and leave the chain. Recalls, misdeliveries, and misreads are considered. We assume that phantom reads are negligible and that multiple reads of the same tag are handled during the data collection process.

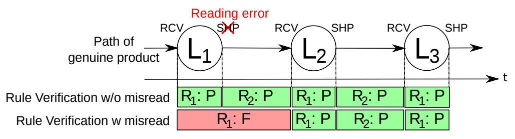
Given the described infrastructure, a counterfeit equipped with a tag and a valid ID will appear as a genuine product (unless human inspection is performed). We therefore define



(a)



(b)



(c)

Fig. 1. Rule verification results for events generated (a) by a genuine product, (b) by both genuine and clone products together, and (c) by a genuine product while considering misreads. P and F stand for *Pass* and *Fail* resp.

a clone as a counterfeit that carries the ID of a genuine product. Counterfeiters, once they have obtained a list of valid IDs, can simply equip their products with tags carrying those IDs. Counterfeiters do not compromise partners or their RFID infrastructure; they can, however, do business with them (e.g., by pretending to be a legal seller). Additionally, we assume that legal partners are not malicious (i.e., they are not, nor do they protect, counterfeiters), but *curious* about each others' activities (e.g., in order to gather strategic information to use during negotiation phases). We also assume that a clone appearing before the genuine product enters, or after it leaves the supply chain, is easily detected by verifying the corresponding events.

III. Clone Detection Mechanism

Our clone detection mechanism is based on verifying the correctness of sequences composed of two time-consecutive events in the tag trace against different rules.

A. Rule Verification

For a given tag, its trace is a collection of events associated to all observed transport processes, i.e., shipping and receiving, in the supply chain. A correct sequence of events is given by two consecutive transport-related events, e_i resp. e_{i+1} (where e_i is the i-th event in a time-sorted trace), that respect one of the following rules:

$$R_1 : e_i = (id, *, L_j, RCV) \rightarrow e_{i+1} = (id, *, L_j, SHP)$$
$$R_2 : e_i = (id, *, L_j, SHP) \rightarrow e_{i+1} = (id, *, L_o, RCV)$$

Rule R_1 says that, for a given tag, a receiving event (RCV) recorded at location L_j must be followed by a shipping event (SHP) recorded at the same location, while rule R_2 says that, for a given tag, a shipping event recorded at location L_j must be followed by a receiving event recorded at a different location L_o. While traveling within the supply chain, a product goes through a series of different locations L_u (that can be directly mapped to the different partners or to different partner installations such as warehouses), where entry events (i.e., a RCV at the in-dock) and exit events (i.e., a SHP at the out-dock) are recorded. A trace containing events generated by one traveling product (assumed to be the genuine product) presents correct sequences of events, e.g., SHP-RCV-SHP-RCV. On the other hand, a trace containing events generated by two (or more) traveling products carrying the same ID (i.e., the genuine product plus its clone(s)) eventually presents incorrect sequences of events[2], e.g., SHP-RCV-RCV-SHP. Fig. 1(a) shows both the path and trace of a genuine product and the consequent rule verification results (all rules successfully verified, i.e., all *pass*), while Fig. 1(b) shows the rule verification results for the same trace but containing events generated by both the genuine product and its clone; since the trace contains incorrect sequences, some rule verifications have as result *fail*.

B. Clone Detection

Ideally, clone detection is straightforward: if a trace only contains correct sequences, we can conclude that there is no clones for the considered ID. Inversely, if a trace contains one or more failed rule verifications, we can conclude that a clone is present. Unfortunately, tag misreads can lead to an incorrect clone evaluation: a missing event can cause either a false failed rule verification or a false passed one, which can lead to false positives (i.e., detecting a non-existent clone) or false negatives (i.e., a clone passed unnoticed). Fig. 1(c) shows the impact of tag misreads on rule verification for false failed verifications.

Therefore, we propose two different clone detection methods that consider (mis)reads (as independent probabilistic events[3]) while evaluating the presence of clones: (i) a rule-verification-ratio-based (RVR) clone detection, and (ii) a binomial-test-based (BT) clone detection.

In (i), the clone detection is based on the ratio R_{RV} between VR_F, the number of verified rules with result *fail*, and VR, the total number of verified rules. VR_F is given by misreads and clones; intuitively, if VR_F exceeds the potential number of failed rule verifications caused by misreads, we can assume the presence of clones. Additionally, considering the same probability of misreads and two traces having the same number of events, the one containing more failed rule verifications gives clearer evidence on the presence of clones. Therefore, if R_{RV} is greater or equal than a certain threshold τ, clones are detected:

$$R_{RV} = \frac{VR_F}{VR} \tag{1}$$

$$\begin{aligned} R_{RV} < \tau &\rightarrow No\ clone \\ R_{RV} \geq \tau &\rightarrow Clone \end{aligned} \tag{2}$$

In (ii), we assume that misreads are read failures occurring with probability p_{mr}; reads are thus considered binomially distributed with failure probability p_{mr}. The BT method tests the deviation from the expected number of failed rule verifications caused by misreads considering the given distribution. If this deviation exceeds a certain value, clones are detected. BT uses the binomial test, where the null hypothesis H_0 says that the failed rule verifications are caused by misreads, while the alternative hypothesis H_A says that those are caused by clones:

$$Pr(K \geq k) = \sum_{k=N_{ME}}^{N_E} \binom{N_E}{k} {p_{mr}}^k (1-p_{mr})^{N_E-k} \tag{3}$$

$$\begin{aligned} Pr(K \geq k) \leq \alpha \rightarrow H_0 \\ Pr(K \geq k) > \alpha \rightarrow H_A \end{aligned} \tag{4}$$

where N_{ME} is the total number of missing events that would restore all the incorrect sequences in the considered trace, N_E is the total number of events, and α is the significance level. N_{ME} is the sum of the number of missing events at each failed rule verification N^u_{ME}; for example, if $e_i = (id, *, L_j, RCV)$ and $e_{i+1} = (id, *, L_o, SHP)$, the number of missing events is two: $(id, *, L_j, SHP)$ and $(id, *, L_o, RCV)$. Table I summarizes combinations of failed rule verification and missing events.

TABLE I
RELATIONS BETWEEN FAILED RULE VERIFICATIONS AND MISSING EVENTS

L_i	S_i	L_j	S_j	Failed Rule	Missing events (N^u_{ME})
A	RCV	A	RCV	R_1	A-SHP, B-RCV, B-SHP (3)
A	RCV	B	RCV	R_1	A-SHP (1)
A	RCV	B	SHP	R_1	A-SHP, B-RCV (2)
A	SHP	A	RCV	R_2	B-RCV, B-SHP (2)
A	SHP	A	SHP	R_2	B-RCV, B-SHP, A-RCV (3)
A	SHP	B	SHP	R_2	B-RCV (1)

IV. Evaluation of the Clone Detection Mechanism

We evaluate our clone detection mechanism by a simulation study. Simulations are based on a custom-built RFID-enabled supply chain simulator, which, from a given supply chain, simulates the flow of products and generates repositories containing the tag events recorded by each partner during the simulation.

For our study, we define a supply chain of 15 partners distributed as a 4-level binary tree (Fig. 2(a)). The manufacturer (1st level) produces 1000 genuine products per day; it equips each product with a tag and a unique identifier and then stocks and consequently ships the products to national wholesalers (2nd level). They stock, repack, and ship products to regional wholesalers (3rd level), who, in turn, stock, repack, and ship products to local retailers (4th level). Retailers store the products on their shelves, which then leave the supply chain at the points of sale. All partners ship products once per day; each partner shares its products equally among its business partners.

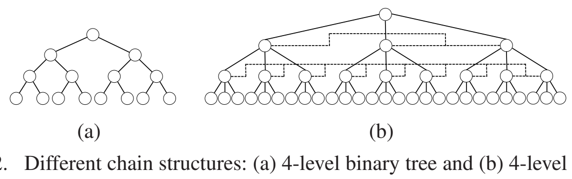
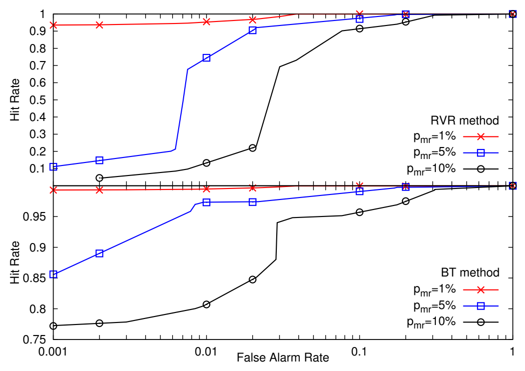


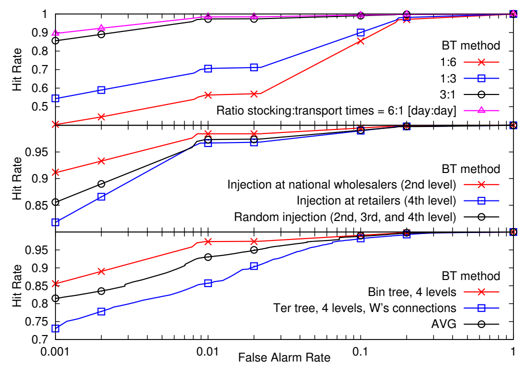
Fig. 2. Different chain structures: (a) 4-level binary tree and (b) 4-level ternary tree with wholesalers connections.

[2]There is still the possibility that events generated by both the genuine and the clones form a correct sequence; we evaluate detection errors in Section IV.

[3]Reads are mainly affected by tags' position, orientation, and density. Since products are moved and possibly replaced between each read, we assume that, for a given tag, a misread at time t_i will not influence the next read at t_{i+1}.



(a)



(b)

Fig. 3. Performance comparison (a) between the RVR (top) and the BT (bottom) methods by varying the misread probability, and (b) performance for the BT by varying the ratio between stocking and transport times (upper graph), the clone injection point (middle), and the supply chain structure (bottom).

Each day, 10 counterfeits (1% of the daily genuine production) are randomly injected into the supply chain at any level (except at the 1st level); each counterfeit carries an ID randomly selected amongst those deployed by the manufacturer. Partners record and store receiving and shipping events related to each observed product. Stocking, transport, and storing times, as well as the misread probability p_{mr}, follow a normal distribution. Table II summarizes the parameters for the described chain.

Fig. 3 shows the evaluation results. Each curve is obtained by averaging 20 simulation runs, each one considering 2 months of production. Clone detection is executed at the points of sale; a trace thus includes all the events for the considered ID until either the genuine or the clone product reaches a point of sale.

In general, the BT method outperforms the RVR method for small false alarm rates ($FAR < 1\%$), while both present similar performance for higher false alarm rates ($FAR > 10\%$). For the described chain (Fig. 3(a) - curve $p_{mr} = 5\%$), for $FAR = 0.1\%$, BT (bottom graph) presents a hit rate equal to 85.5%, while RVR (top graph) equal to 11%. Those values increase up to 97.5% and 74.5% respectively for $FAR = 1\%$, and to 99.8% and 97% respectively for $FAR = 10\%$.

Fig. 3(a) shows the impact of the misread probability on both BT and RVR: a high p_{mr} increases the uncertainty regarding the cause of rule verification failures (clone or misreads), leading to performance degradation, while a small p_{mr} leads to a near-ideal situation in which rule verification failures are only due to clones, and therefore, to an easier detection.

Fig. 3(b) shows the impact of the ratio between stocking and transport times, the supply chain structure, and the counterfeit injection point on the BT method. A shorter stocking time reduces the probability of finding a clone's events between two genuine RCV and SHP (reducing R_1 failures), while a longer transport time increases the probability of finding a clone's events between two genuine SHP and RCV (reducing R_2 failures): the smaller the ratio between stocking and transport times, the more clones may pass unnoticed (Fig. 3(b), upper graph). Injecting clones into a lower level of the chain generates fewer events than injecting clones into an upper level; this produces less evidence regarding the presence of clones, increasing the probability that those pass unnoticed (Fig. 3(b), middle graph). For the same number of incorrect sequences and p_{mr}, smaller traces give clearer evidence on the presence of clones than longer traces. Fig. 3(b) (bottom graph) shows the impact of the trace length by considering a 4-level binary tree (average trace length equal to 6.02 events) and a 4-level ternary tree with wholesalers connections (Fig. 2(b), average trace length equal to 12.65). We also consider several alternative structures by varying the number of levels, partners per level, and connections among partners (curve AVG shows an average of those structures' performance). We note that structures that present a similar average trace length have similar performance.

TABLE II
SIMULATION PARAMETERS FOR THE CONSIDERED SUPPLY CHAIN

Parameter	Value
Misread probability	Normal distribution, μ=5%, σ=1%
Genuine products production rate	1000/day
Clone products production rate	10/day
Product production time	2 months
Shipping time	1/day at 8AM
Stocking time	Normal distr., μ=3 days, σ=12 hours
Transport time	Normal distr., μ=1 day, σ=6 hours
Output load	Equally distributed
Supply chain structure	15 partners, 4-level binary tree
Counterfeit injection point	Random

V. Cryptographic Toolkit

In this section, we briefly introduce the cryptographic tools deployed in our privacy-preserving protocols.

ElGamal Cryptosystem. The ElGamal cryptosystem [12] is an asymmetric encryption scheme based on the discrete log problem. The private key is x and the public key is $y = g^x$ (with generator g). A message m is encrypted as $E_y(m) = (g^r, m \cdot y^r)$, where r is a randomly chosen element. This scheme is multiplicatively homomorphic, i.e., $E_y(m_1) \cdot E_y(m_2) = E_y(m_1 \cdot m_2)$. To enable additive homomorphism, where $E_y(m_1) \cdot E_y(m_2) = E_y(m_1 + m_2)$, m is encrypted as $(g^r, g^m \cdot y^r)$. The ElGamal cryptosystem is both semantically secure and malleable, which enables ciphertext randomization: Alice can, by adding an encryption of 0 to a ciphertext (encryptions under the same y, but different r), make it indistinguishable from other ciphertexts even to Bob who originally encrypted the message.

Distributed Key Generation. The ElGamal cryptosystem can be converted into a threshold protocol using distributed key generation protocols; the private key is then shared among n players so that at least t out of the n players are needed to decrypt a ciphertext. We base our distributed key generation on the protocol proposed by Pederson [13].

Oblivious Transfer. Oblivious transfer (OT) [14] is a protocol that enables Alice to retrieve a given piece of information offered among alternatives by Bob, without Bob knowing which piece was chosen. In a *1-out-of-2* OT, denoted as OT_2^1, Bob has two messages m_0 and m_1; the OT_2^1 allows Alice to choose and receive exactly one of m_0 and m_1 without Bob knowing which one Alice selected. In our protocol, we use the constant-round OT proposed by Naor and Pinkas [15].

Yao Circuits. The Yao scheme was introduced by Yao [16] and described thoroughly by Lindell and Pinkas [17]. It is a two-player protocol for securely computing any given function represented as a boolean circuit: given a circuit C that takes as input two bit strings z_1 and z_2 and outputs $C(z_1, z_2) \in 0, 1$, its *garbled* version C' allows Alice and Bob, owning z_1 and z_2 respectively, to compute $C(z_1, z_2)$ without revealing each other's input. To create the garbled circuit, Alice assigns to each wire of C two random keys, which map to binary 0 and 1 respectively. For each gate in C, Alice redefines its truth table according to this mapping. Alice then sends C' to Bob, along with the keys corresponding to her input bits, and the mapping between the keys of each C's output wire and the value represented by these keys. For each of Bob's input bits, Alice and Bob perform an OT_2^1, allowing Bob to retrieve the corresponding key. Bob then evaluates the circuit, obtaining a key for each of C's output wires; using the key/value mappings previously received, he determines (*opens*) and then shares the result of the computation. In our protocols, we keep the result of the evaluation secret by mapping the keys of C's output wires to encrypted values instead of bits (e.g., $E_y(0)$ instead of 0), and use the scheme proposed by Malkhi et al. [18].

Mixnets. Mixnets are multistage systems used to anonymize the origin of data through cryptography and permutations [19]. Informally, a mixnet is a protocol that takes as input a list of ciphertexts and gives as output a new, random list of ciphertexts which presents a one-to-one correspondence between the underlying plaintexts of input and output items. For our protocol, we perform a mixnet on a matrix M of encrypted items based on the presented additively homomorphic ElGamal cryptosystem. Each server first mixes M by choosing and applying a random permutation M_P, where $M' = M_P \cdot M \cdot M_P^{-1}$ corresponds to swapping rows and columns of M according to M_P. Then, it randomizes M' by adding an encryption of 0 to each $M'(i, j)$ as $E_y(M'(i, j) + 0) = E_y(0) \cdot E_y(M'(i, j))$.

VI. Privacy-preserving Clone Detection

We now present the privacy-preserving BT and RVR methods, obtained through secure multi-party computation protocols not relying on trusted third parties.

A. Security and Privacy Requirements

Our protocols aim at enabling the private evaluation of the presence of clones for a given ID in a scenario where n partners P_i own N_{TE} tag events e_j composing the trace of ID. During the execution of the protocols, a partner P_o will not learn anything about other partner events and relationships, and neither about the structure of the supply chain. This includes not learning an event itself, as well as information like "event e_q follows event e_w in time", or "partner P_e owns an event that follows partner P_r's event". All the information that partners know before the execution of the protocols and that can be learned from the outputs of the protocols is not considered sensitive. We assume that N_{TE}, n, and the detection mechanism are known to the partners. Given the scenario described in Section II, in which partners are not involved in counterfeiting activities, we consider protocols that are secure in the semi-honest adversary model; partners are interested in the correct execution of the protocols in order to detect clones, but they are still *curious* about other partners' activities.

B. Proposed Protocols

Privacy-preserving RVR verifies rules R_1 and R_2 for each pair of events, assigning 0 if at least one of those is respected, and 1 otherwise. Then, it identifies pairs composed of two time-consecutive events, and finally, it selects and sums only the rule verification results associated to the identified pairs, obtaining VR_F, the number of verified rules with result *fail*.

Similarly, privacy-preserving BT verifies rules R_1 and R_2 for each pair of events, and evaluates the number of missing events for each considered pair. Then, it identifies pairs composed of two time-consecutive events, and finally, it selects and sums only the numbers of missing events associated to the identified pairs, obtaining N_{ME}, the total number of missing events.

We assume that before starting the protocols, all partners (players) obtain the necessary information to establish a connection to each others, e.g., through the DS, which knows who owns events on a certain ID, and can then bootstrap the network upon a partner's request for detecting clones of ID.

Privacy-preserving RVR.

1) All players jointly generate a public key y and a shared secret key x for an instance of the additively homomorphic ElGamal cryptosystem using the distributed key generation scheme as described above.
2) For all pairs of events (e_i, e_j), $i \neq j$, the players owning events e_i and e_j jointly evaluate $E_y(\text{GT}(e_i, e_j) := [e_i(T) > e_j(T)])$ and $E_y(\text{GT}(e_j, e_i))$ using a Yao circuit with $E_y(1)$ and $E_y(0)$ mapped to the output wire representing 1 ($e_i(T) > e_j(T)$) and 0 ($e_i(T) < e_j(T)$) respectively. After the circuit evaluation, the player owning e_i opens the output, obtaining an encryption under y of $\text{GT}(e_i, e_j)$. Then, he computes $E_y(\text{GT}(e_j, e_i)) = E_y(1 - \text{GT}(e_i, e_j)) = E_y(1) \cdot E_y(\text{GT}(e_i, e_j))^{-1}$ and sends $E_y(\text{GT}(e_j, e_i))$ to the owner of e_j.

3) For all pairs of events (e_i, e_j), $i \neq j$, the players owning events e_i and e_j jointly evaluate $E_y(\textsc{rv}(e_i, e_j))$, which corresponds to the rule verification on the pair of events (e_i, e_j), using a Yao circuit with $E_y(1)$ and $E_y(0)$ mapped to the output wire representing 1 (failed rule verification) and 0 (passed rule verification) respectively. The player owning e_i opens the output of the circuit, and obtains an encryption under y of $\textsc{rv}(e_i, e_j)$ (which is then not shared with the owner of e_j).
4) For each of his events e_i, each player adds the bits he got in step 2, obtaining $E_y(\textsc{rk}(e_i))$, which corresponds to the position (i.e., the rank) of e_i in a time-sorted list of all considered N_{TE} events. Each player performs this operation locally, by calculating $E_y(\textsc{rk}(e_i)) = E_y(\sum_{j \neq i} \textsc{gt}(e_i, e_j)) = \prod_{j \neq i} E_y(\textsc{gt}(e_i, e_j))$. We can now imagine an N_{TE}-by-N_{TE} matrix formed by arranging the encrypted $\textsc{rv}$ in the matrix and filling the diagonal with the encrypted $\textsc{rk}$:

$$M = \begin{pmatrix} E_y(\textsc{rk}(e_1)) & \dots & E_y(\textsc{rv}(e_1, e_{N_{TE}})) \\ \vdots & \ddots & \vdots \\ E_y(\textsc{rv}(e_{N_{TE}}, e_1)) & \dots & E_y(\textsc{rk}(e_{N_{TE}})) \end{pmatrix}$$

Each line i of M relates to an event e_i, the element $M(i,i)$ to the rank of e_i, and the element $M(i,j)$ to the result of the rule verification between e_i and e_j.
5) The players cooperatively perform a mixnet over M. To perform this, all players send the encryptions obtained in steps 2 and 4 to the player P_0, who then arranges them into the matrix M. The mixnet can be executed as described above: each player in turn, starting with P_0, randomly permutes the matrix, randomizes each element $M(i,j)$, and sends it to the next player. We denote the final result of the mixnet by M_n.
6) The players collaboratively decrypt the diagonal of M_n; it is now possible to identify two time-consecutive events, and therefore to identify and sum the corresponding rule verification results: if event e_i has rank u ($M_n(i,i) = u$) and event e_j has rank $u + 1$ ($M_n(j,j) = u + 1$), then the correspective rule verification result for that pair is at $M_n(i,j)$. Summing all the identified rule verification results gives an encryption under y of the number of failed rule verifications VR_F.
7) The players collaboratively decrypt $E_y(VR_F)$. Each partner P_i, based on its local threshold τ, can now evaluate the presence of clones through (1) and (2).

Privacy-preserving BT. The privacy-preserving BT follows the RVR protocol. We point out distinctions in the following steps:

3) For all pairs of events (e_i, e_j), $i \neq j$, the players owning events e_i and e_j jointly evaluate $E_y(\textsc{me}(e_i, e_j))$, corresponding to the number of missing events for the pair of events (e_i, e_j), using a three-output Yao circuit. For the three outputs, we map $E_y(0)$ to 0, while we map $E_y(3)$, $E_y(2)$, and $E_y(1)$ to 1 for the first, second, and third output respectively. $E_y(3)$, $E_y(2)$, $E_y(1)$, and $E_y(0)$ represent 3, 2, 1, or 0 missing events respectively (Table I). The player owning e_i opens the outputs of the circuit, and sums them as $E_y(\sum_{k=1}^{3} N_{ME,k}) = \prod_{k=1}^{3} E_y(N_{ME,k})$, obtaining an encryption under y of $\textsc{me}(e_i, e_j)$ (which is then not shared with the owner of e_j).
4) The N_{TE}-by-N_{TE} matrix M becomes:

$$M = \begin{pmatrix} E_y(\textsc{rk}(e_1)) & \dots & E_y(\textsc{me}(e_1, e_{N_{TE}})) \\ \vdots & \ddots & \vdots \\ E_y(\textsc{me}(e_{N_{TE}}, e_1)) & \dots & E_y(\textsc{rk}(e_{N_{TE}})) \end{pmatrix}$$

Each line i of M relates to an event e_i, the element $M(i,i)$ to the rank of e_i, and the element $M(i,j)$ to the number of missing events between e_i and e_j.
6) After collaboratively decrypting the ranks and identifying pairs of time-consecutive events, each player sums all the corresponding numbers of missing events, obtaining an encryption under y of the total number of missing events N_{ME}.
7) The players collaboratively decrypt $E_y(N_{ME})$. Each partner P_i, based on its local significance level α and misread probability p_{mr}, can now evaluate the presence of clones through (3) and (4).

VII. Evaluation of the Privacy-preserving Clone Detection

A. Security and Privacy Guarantees

We assume that the adversary is computationally bounded, semi-honest, and that controls less than t players. The proposed protocols guarantee privacy against $t - 1 < n$ compromised players, where t is both the threshold used for the (t, n)-sharing of the ElGamal secret key and the number of mixnet servers ($t = n$ in the proposed protocols).

A Yao circuit is private when an honest player engages in a computation with a semi-honest adversary. In the proposed protocols, two players P_i and P_j can securely compute $\textsc{rv}(e_i, e_j)$, $\textsc{me}(e_i, e_j)$, and $\textsc{gt}(e_i, e_j)$ without revealing each other's input. When player P_i evaluating the circuit opens the output, he cannot distinguish between the possible values of the result (e.g., 0 or 1 for $\textsc{rv}$), since this has been encrypted under the ElGamal key by P_j. Moreover, P_i does not send the result to P_j, who cannot then learn anything from the circuit evaluation.

The mixnet disassociates the data from the players in a way that information about the time-ordering of events can be made public, which does not reveal any information if the events themselves are encrypted and unlinkeable to players.

B. Performance Evaluation

We evaluated our privacy-preserving clone detection mechanism for a scenario in which n partners P_i want to run a clone detection over N_{TE} distributed events by considering round and bit complexity, execution times, and bandwidth consumption.

Complexity Estimation. All pairwise Yao circuit evaluations can be performed in parallel and, for a single evaluation, with the participation of only the players who own the two considered events. ElGamal distributed key generation is constant-round, while the mixnet requires n rounds. Therefore, our protocols have round complexity $\mathcal{O}(n)$.

Asymptotically, the bit complexity is dominated by the Yao circuits and by the mixnet. Assuming event fields with constant bit sizes, the size of the Yao wire keys (80 bits [18]) smaller than the size l of the OT key (same size as for the ElGamal encryption), and the N_{TE} events equally distributed among the n partners, the bit complexity for the Yao circuits is $\mathcal{O}(l \cdot {N_{TE}}^2 \cdot (1 - 1/n))$. For the mixnet, at each stage an (ElGamal) encrypted matrix of size ${N_{TE}}^2 \cdot l$ bits is sent, thus giving a complexity of $\mathcal{O}(n \cdot l \cdot {N_{TE}}^2)$. Therefore, the total bit complexity is $\mathcal{O}(l \cdot {N_{TE}}^2 \cdot (n - 1/n + 1))$.

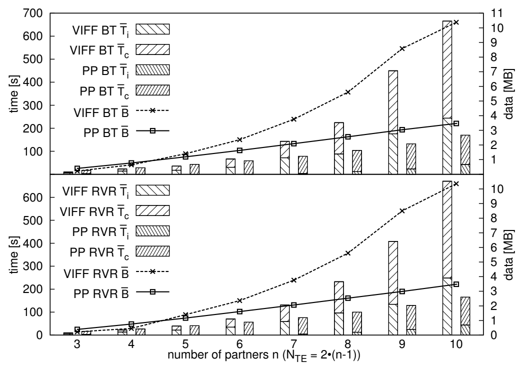
Experimental Evaluation. Within our experimental evaluation, we denote $\bar{T}_w$, $\bar{T}_c$, $\bar{T}_i$, and $\bar{B}$ as the average wall clock time per player, the average computation time per player, the average idle time per player, and the average bandwidth consumption (data sent) per player, respectively (for P_i, $T_{w,i} = T_{c,i} + T_{i,i}$).

We implement and evaluate according to the defined metrics our privacy-preserving (PP) RVR and BT protocols. For performance comparison, we additionally implement and evaluate both RVR and BT methods within two secure multi-party computation (SMC) frameworks: VIFF [10] and Sharemind [9][4]. VIFF provides secure arithmetic operations and comparisons on shared values in a peer-to-peer model (all players participate in the computation). Sharemind uses additive secret sharing schemes and composable protocols for arithmetic operations, bitwise operations, and comparisons in a client-server model: clients use secret sharing to split private data between several servers, which then run a given function on those shares. Sharemind supports three servers. Both frameworks offer protection against a semi-honest adversary corrupting up to $\lfloor N/2 \rfloor$ computation players. VIFF considers computationally-bounded adversaries, while Sharemind computationally-unbounded ones.

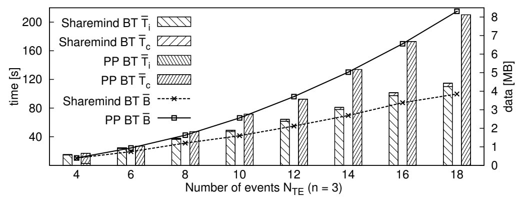
Fig. 4 shows the evaluation results for our PP RVR and BT implementations against (a) VIFF and (b) Sharemind implementations. We run VIFF (v.0.7.1), Sharemind (v.1.81), and our implementations in a network of identical, dual-CPU 3 GHz Pentium 4 machines, with 1 GB of memory and connected via a LAN. Each plot is obtained by averaging 20 runs. The group/key length l for the OT and ElGamal encryption is 1024 bits. The size of event fields T and L is 32 bits, while S is 1 bit.

Fig. 4(a) shows the performance of our PP RVR and BT implementations against VIFF implementations within a real-world scenario where $N_{TE} = 2 \cdot (n - 1)$ (i.e., in which each partner has two events, except the manufacturer and the retailer who have just one event). We note that the implementations of RVR and BT present similar performance, while both our PP RVR and BT implementations outperform VIFF ones for $n > 5$ (for $n = 10$, they reduce $\bar{T}_w$ by 75% and $\bar{B}$ by 66%) and present similar results for smaller n. The impact of a small number of partners is also visible in Fig. 4(b), where, for $n = 3$ and varying N_{TE}, the Sharemind BT implementation outperforms our PP BT one. Note that both frameworks present a security level of $\lfloor n/2 \rfloor$; our protocols provide stronger security $(n - 1)$, which requires additional computational efforts.

[4]We also considered other frameworks such as FairplayMP [20], SMCL [21], SCET [22], and P4P [23]. Due to unavailable source code, scarce support, and/or limited operations, they were not suitable for our purposes.



(a)



(b)

Fig. 4. Performance comparison (a) between our privacy-preserving (PP) BT and RVR implementations and VIFF BT and RVR implementations, and (b) between our PP BT implementation and Sharemind BT implementation.

Although SMC frameworks could implement secure computations for any probabilistic polynomial-time function, they can be impractical in many relatively complex scenarios (e.g., in which $n > 5$). Our protocols are optimized for a peer-to-peer model where a relatively high number of partners securely compute a specific function. Mainly, the improved performance is obtained by dividing the computation load into several pairwise computations, rather than have all players participate in each computation (as in VIFF). Differently, Sharemind operates in a client-server model with a small number of computing servers; performance of our protocols cannot benefit from pairwise computations within that scenario.

Although the experimented wall clock times are relatively high, we note that the computation time can be reduced by more powerful hardware, while, by applying an appropriate scheduling, the idle time (which impact can increase for partners communicating over Internet) can be used to evaluate other tags. Additionally, we could improve the performance of our protocols by reducing their security level to $\lfloor n/2 \rfloor$.

VIII. Related Work

Trace-based solutions for clone detection were initially conceived for the pharmaceutical industry; Kuh et al. [24] suggested to record track-and-trace data of tagged pharmaceutical products in order to create drug pedigrees. Staake et al. [3] discussed the deployment of track-and-trace solutions with plausibility checks to detect counterfeits within EPCglobal networks [2], pointing out the negative effect of incomplete drug pedigrees due to partners not recording or storing tracing data. Lehtonen et al. [25], [26] explored trace-based clone detection from incomplete traces through machine learning techniques. The authors considered incomplete traces caused by both tag misreads and partners not sharing tag observations.

Both works present effective detection performance, but require a training step (or a global knowledge of supply chain structures and products flow) that makes them less robust to supply chain dynamics, recalls, and misdeliveries. Mirowski et al. [27] presented a general approach for detecting change of tag ownership (occurring when stealing or cloning a tag) by analyzing tag traces for anomalous behaviors using the principles of intrusion detection. The system does not need any predefined correct behavior, but it is prone to false positives. All these solutions assume either a central repository that stores all tag observations or that partners share tag observations in plaintext; given the concerns that partners have regarding the possible use of tag observations to infer sensitive information, none of the proposed solutions may be directly applicable in a real-world scenario. Additionally, few of them consider tag misreads or are robust to supply chain dynamics, recalls, and misdeliveries.

The literature includes several cryptographic primitives for implementing secure multi-party computation (SMC) protocols [28]: oblivious transfer [14], [15], homomorphic encryption [12], garbled circuit [16], [17], and secret sharing [29]. Generic constructions that implement SMC for any probabilistic polynomial-time function exist (e.g., the BMR protocol [30]), but they can be very impractical due to their complexity. Examples of SMC frameworks based on generic constructions are Fairplay [18], FairplayMP [20], Sharemind [9], VIFF [10], SMCL [21], SCET [22], and P4P [23]. Ad-hoc constructions that implement SMC for specific functions are often more efficient, but must be designed specifically for each function. In the past years, ad-hoc constructions have been proposed for genomic computation [31], data mining [32], surveys [33], auctions [34], and remote diagnostics [35]. In the context of supply chains, Atallah et al. [36], [37] proposed several privacy-preserving protocols for capacity allocation, e-auctions, and collaborative planning, forecasting, and replenishment.

IX. Conclusion

We presented a novel, privacy-preserving, trace-based clone detection mechanism for RFID-enabled supply chains which enables supply chain partners to detect counterfeits without revealing information on their private tag-related data. We evaluated the proposed mechanism through a simulation study, proving its effectiveness under several conditions. We designed and implemented ad-hoc secure multi-party computation (SMC) protocols for executing privacy-preserving clone evaluation, and showed their performance improvements with respect to similar implementations within existing SMC frameworks. Future work may address different adversary models for both counterfeiters and partners, as well as possible additional performance improvements for the privacy-preserving implementations.

Acknowledgment

The authors would like to thank Basil Hess for the design and implementation of the RFID-enabled supply chain simulator.

References

[1] Organisation for Economic Co-operation and Development, *The economic impact of counterfeiting and piracy*. OECD, June 2008.

[2] EPCglobal, “Architecture framework v. 1.3,” Standard, 2009.

[3] T. Staake, F. Thiesse, and E. Fleisch, “Extending the EPC network: the potential of RFID in anti-counterfeiting,” in *Proc. ACM SAC*, 2005.

[4] K. Pramatari, “Collaborative supply chain practices and evolving technological approaches,” *Supply Chain Management*, vol. 12, no. 3, 2007.

[5] M. Lehtonen, T. Staake, F. Michahelles, and E. Fleisch, “From identification to authentication - a review of RFID product authentication techniques,” Printed handout of RFIDSec06, 2006.

[6] S. Spiekermann and S. Evdokimov, “Privacy enhancing technologies for RFID - a critical investigation of state of the art research,” in *Proc. IEEE Privacy and Security*, 2009.

[7] T. Burbridge, V. Broekhuizen, J. Farr, D. Zanetti, E. García Muñoz, J. J. Cantero, M. A. Guijarro, and J. Baños, “RFID network confidentiality,” BRIDGE deliverable D-4.5.1, 2007.

[8] R. H. Clarke, D. Twede, J. R. Tazelaar, and K. K. Boyer, “Radio frequency identification (RFID) performance: the effect of tag orientation and package contents,” *Packag. Technol. and Sci.*, vol. 19, no. 1, 2006.

[9] D. Bogdanov, S. Laur, and J. Willemson, “Sharemind: a framework for fast privacy-preserving computations,” in *Proc. ESORICS*, 2008.

[10] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen, “Asynchronous multiparty computation: Theory and implementation,” in *Proc. Int. Conf. on Practice and Theory in Public Key Cryptography*, 2009.

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

[12] T. El Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *Proc. CRYPTO 84*, 1985.

[13] T. P. Pederson, “A threshold cryptosystem without a trusted party (extended abstract),” in *EUROCRYPT*, 1991.

[14] M. Rabin, “How to exchange secrets by oblivious transfer,” Aiken Computation Laboratory, Harvard University, Tech. Memo TR-81, 1981.

[15] M. Naor and B. Pinkas, “Efficient oblivious transfer protocols,” in *Proc. ACM-SIAM Sym. on Discrete Algorithms*, 2001.

[16] A. C.-C. Yao, “How to generate and exchange secrets,” in *Proc. Sym. on Foundations of Computer Science*, 1986.

[17] Y. Lindell and B. Pinkas, “A proof of Yao’s protocol for secure two-party computation,” *Electronic Colloq. on Comput. Compl.*, no. 063, 2004.

[18] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, “Fairplay - a secure two-party computation system,” in *Proc. USENIX Security Sym.*, 2004.

[19] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Communications of the ACM*, vol. 24, no. 2, 1981.

[20] A. Ben-David, N. Nisan, and B. Pinkas, “FairplayMP: a system for secure multi-party computation,” in *Proc. ACM CCS*, 2008.

[21] http://www.brics.dk/SMCL/.

[22] http://www.sikkerhed.alexandra.dk/uk/projects/scet.htm.

[23] Y. Duan, J. F. Canny, and J. Z. Zhan, “Efficient privacy-preserving association rule mining: P4P style,” in *CIDM*, 2007.

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

[25] M. Lehtonen, F. Michahelles, and E. Fleisch, “Probabilistic approach for location-based authentication,” in *Proc. IWSSI*, 2007.

[26] ——, “How to detect cloned tags in a reliable way from incomplete RFID traces,” in *Proc. IEEE Int. Conf. on RFID*, 2009.

[27] L. Mirowski and J. Hartnett, “Deckard: a system to detect change of RFID tag ownership,” *IJCSNS*, vol. 7, no. 7, 2007.

[28] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[29] A. Shamir, “How to share a secret,” *Comm. ACM*, vol. 22, no. 11, 1979.

[30] D. Beaver, S. Micali, and P. Rogaway, “The round complexity of secure protocols,” in *Proc. ACM Sym. on Theory of computing*, 1990.

[31] S. Jha, L. Kruger, and V. Shmatikov, “Towards practical privacy for genomic computation,” in *Proc. IEEE Sym. on Secur. and Priv.*, 2008.

[32] Y. Lindell and B. Pinkas, “Secure multiparty computation for privacy-preserving data mining,” Cryptology ePrint Archive, Report 197, 2008.

[33] J. Feigenbaum, B. Pinkas, R. Ryger, and F. Saint-Jean, “Secure computation of surveys,” *EU Work. on Secure Multiparty Protocols*, 2004.

[34] M. Naor, B. Pinkas, and R. Sumner, “Privacy preserving auctions and mechanism design,” in *Proc. Conf. on Electronic Commerce*, 1999.

[35] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel, “Privacy-preserving remote diagnostics,” in *Proc. ACM CCS*, 2007.

[36] M. Atallah, H. Elmongui, V. Deshpande, and L. Schwarz, “Secure supply-chain protocols,” in *Proc. IEEE Int. Conf. on E-Commerce*, 2003.

[37] M. J. Atallah, M. Blanton, V. Deshpande, K. Frikken, J. Li, and L. Schwarz, “Secure collaborative planning, forecasting, and replenishment (SCPFR),” *M&SOM*, 2005.