

Privacy-Preserving Pattern Matching for Anomaly Detection in RFID Anti-Counterfeiting

Florian Kerschbaum¹ and Nina Oertel^{1,2}

¹ SAP Research, Karlsruhe, Germany
florian.kerschbaum@sap.com
nina.oertel@sap.com

² Chair of Business Administration and Information Systems,
University of Mannheim, Germany

Abstract. Traces of RFID-equipped item can be used to detect counterfeits. Nevertheless companies are reluctant to share the necessary traces, since it is unclear what can be inferred from them. In this paper we present a provably secure pattern matching algorithm that can be used for distributed anomaly detection. We improve performance and detection capabilities compared to competing approaches by storing partial, malleable information on the RFID tag.

1 Introduction

Counterfeit products lead to huge financial losses for legally run business. For example, European Customs seize up to one hundred million counterfeit articles per year [6]. It is well-known that RFID event traces can be used for anti-counterfeiting [15, 24, 19, 26]. Yet companies are still reluctant to share the necessary data, since it is unclear what other information can be inferred from it [13, 22].

Cryptography offers the ultimate solution: Secure Multi-Party Computation (SMC) [3, 9, 25]. In SMC a number of parties compute a joint function on their combined inputs without revealing any additional information except the result. Since general SMC allows the computation of any function, this function could be the correlation algorithm. Du and Atallah [5] have first proposed this setup.

SMC can be used for rather simple, infrequently used correlations [26], but it is still prohibitively slow for large-scale problems. The first measurements show a performance penalty compared to non-private computation on the order of tens of thousands [11, 12] and even specialized protocols such as [26] only scale to a few clients. Implementations of privacy-preserving event correlation [14, 16, 17, 20] therefore suggest alternative techniques. These techniques commonly rely on revealing the

information necessary for the detection algorithm while revealing as little as possible additional information.

In this paper we use a different approach. We present a provably secure algorithm for a function with limited privacy. Our algorithm implements pattern matching which can be used as the building block for anomaly detection. We complement the secure protocol by storing partial, malleable (by the attacker) information on the RFID tag. We experimentally evaluate the performance of our algorithm and it is acceptable for the intended use case.

In summary this paper contributes a privacy-preserving pattern matching algorithm that

- is *provably secure* and reveals no information if the pattern does not match.
- can be used to *implement anomaly detection*, e.g. for anti-counterfeiting.
- is *efficient* and can detect a counterfeit in less than 1 second using 20 KBytes of network communication in our use case example.

The remainder of the paper is structured as follows. Related work is reviewed in Section 2. The explanation of our anomaly detection algorithm for anti-counterfeiting follows in Section 3. We then continue by describing the privacy-preserving pattern matching algorithm in Section 4. We evaluate the computation and communication performance of this system in Section 5. Our conclusions are presented in Section 6.

2 Related Work

2.1 RFID for Anti-Counterfeiting

The idea of our use case that RFID event data can be used for anti-counterfeiting has been first suggested in [24]. The first algorithm to detect changes in the owner (attached item in our case) has been presented in [18]. It is purely an anomaly-based detection approach and assumes a central repository of events. A refinement to deal with incomplete traces based on a stochastic detection approach is presented in [15]. The algorithm learns the transition probability from one event to another and identifies low probability transitions. The approach used in this paper based on an evaluation of complete traces was first presented in [19].

The first privacy-preserving RFID counterfeit detection algorithm is presented almost concurrently with this paper in [26]. We improve over this approach in two aspects: First, we enhance detection capabilities by the ability to detect more patterns. Instead of only two types of events –

send and receive – we support an arbitrary number of events. Our detection algorithm has been independently evaluated in [19]. This enhanced capabilities make our algorithm also more general and applicable to related problems in distributed anomaly detection.

Second, we significantly improve performance. In particular, we do not use heavy weight secure computations in order to compute the ordering of events. Instead we store this information on the RFID tag, but ensure that in case the attacker maliciously modifies this information we are still able to detect the pattern (w.h.p.). Our numbers show better performance for a significantly increased case study.

2.2 Privacy in Distributed Intrusion Detection

This paragraph provides an overview on protecting privacy in distributed intrusion detection. The first proposal for a practical system was made in [16]. It introduces the model with a central correlation agent also used in this paper. Privacy is achieved by cleverly pseudonymizing sensitive fields. Further pseudonymizing techniques are given in [14]. An implementation based on Bloom filters as pseudonymizing technique is described in [17, 20].

In [1] key-word based aggregation using encryption and SMC has been implemented. They split the central correlation agent into two mutually distrustful ones. The first called proxy anonymizes the data and the second called database computes the correlation on the anonymized plaintext. This can only be done if the plaintext does not reveal sensitive information. We emphasize that their algorithms are not meant to be run on-line for each event.

In [4] SMC has been implemented for detecting frequent events using entropy and counters. They report running times on the order of minutes and communication on the order of several MBytes and claim an improvement of a factor of roughly 1000 over general frameworks, such as FairPlayMP [2]. No figures are given with respect to non-private computation, but we see a similarity in functionality and reported performance to [11, 12]. Their algorithms use locally pre-aggregated events as input and are therefore also not run on-line for each event.

Efficient protocols for the two-party case of our pattern matching algorithm are given in [10]. The two-party case is significantly simpler, not only because of the limited number of participants, but also, since the pattern is not distributed, it does not need to be sorted. We extend that to the multi-party case and are significantly more efficient.

The advantages an attacker might have from a centralized detection system despite privacy protection are described in [21]. This corresponds to the common problem of privacy-preserving computation that the result may still reveal sensitive information. We recommend to treat detected events – counterfeits – with the necessary care and confidentiality.

3 Anomaly Detection for Anti-Counterfeiting

Radio Frequency Identification (RFID) enables tracking individual products through the supply chain [23]. An RFID tag with a unique identifier (UID) is attached to each item and captured at distinct read points within companies handling the item. A suitable reading device is used to read the UID and a corresponding event is stored in a local database of the company. By default, each company has only access to RFID event data that was captured by readers belonging to the organization.

An event in our algorithm corresponds to reading an RFID tag. When a company X_k processes an item with attached tag with UID id it creates an event with $e.y_j = k$ (i.e. the event type is the organization’s identity). Recall that X_k reads the event number j from the tag. As the item is forwarded along the supply chain, different companies create events for the same tag. Our pattern matching algorithm is applied to an event trace t for a specific tag with UID id . The correlation agent therefore initially sends id to the event sources.

We will now describe the different possibilities of counterfeiters to distribute fakes and the consequences of these actions on the event traces. Suppose all items of a certain type are equipped with UIDs. The first challenge for the counterfeiter is thus to obtain UIDs for the counterfeited goods. One option is to put no identifier at all on the item, but this strategy is easily detectable during authentication. For actually obtaining a UID, the options include guessing random numbers, transferring the UIDs of genuine products to counterfeits or copying the UIDs found on genuine products. For transferring UIDs, counterfeiters may remove (steal) RFID tags from genuine items and reapply them to counterfeits, or they may seek access to UIDs of disposed products and reuse the tags. As a consequence, the UID on a counterfeit product will be either duplicated (in case of copying) or truly unique (for transfer and most probably guessing). Furthermore, the UID will either be valid (for copying and transferring), meaning that a genuine item carrying the same UID exists, or invalid (for guessing). Any UID found on a counterfeit thus has at least one of these properties: it is invalid, has been transferred or is duplicated.

Besides obtaining UIDs, a counterfeiter must distribute the counterfeits and put them on the market, choosing a suitable location and time. Counterfeits can be distributed through illicit supply chains or injected in licit supply chains. Examples of illicit distribution are the smuggling of goods through customs and the sale on flea markets. Selling counterfeits in online shops is another increasingly popular distribution strategy. Counterfeiters also misuse the licit supply chain, sometimes mixing counterfeits with genuine products to better disguise them. For the resulting trace of items it is most relevant whether the chosen channels are visible, i.e. readers are deployed and item movements are captured, or invisible. Illicit distribution channels are likely to contain no read points and thus be invisible, while licit channels can be assumed to be visible.

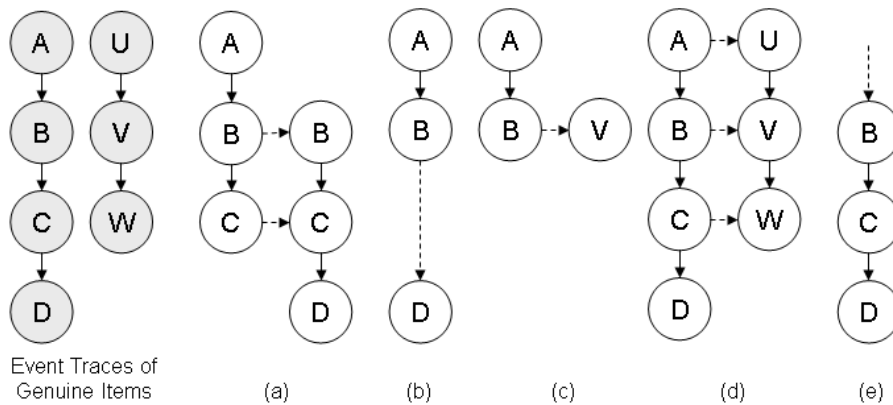


Fig. 1. Consequences of counterfeiter strategies on trace data

In case a counterfeit carries a transferred UID (valid and unique), the events in the trace were triggered by the movements of two items: First by the genuine item until its UID was removed, then by the counterfeit carrying the stolen UID. Up to the transfer point, the trace will be that of a genuine item. When the counterfeit is re-injected in the licit supply chain, the sequence of events will only be valid if the counterfeit directly replaces a genuine item. If the injection takes place further upstream (Figure 1 part (a)), downstream (Figure 1 part (b)) or in another branch of the supply chain (Figure 1 part (c)), the trace will not conform to the traces of genuine items. The resulting traces contain transitions that are

forbidden for genuine items, e.g., the transition $B \rightarrow B$ in Figure 1 part (a).

If counterfeits with duplicate UIDs are injected in the supply chain, the trace that is retrieved for the UID is a mix of all sub-traces created by the multiple items carrying the same UID (Figure 1 part (d)). If items with copied UIDs are injected in the supply chain, this will result in an invalid global trace that contains transitions between events created for different items, albeit they carry the same UID. For example, let event A be triggered by a genuine item, and let it be followed by event U , triggered by a counterfeit with a copied UID. The resulting transition from $A \rightarrow U$ is not allowed for genuine items as they would never travel on that path.

If counterfeits with guessed, i.e. invalid, but unique, UIDs are injected in the supply chain, the trace will be incomplete unless the counterfeiter manages to inject it at a licit producer. In all other cases, the trace starts with an invalid first event, e.g., B (Figure 1 part (e)) which is not allowed for genuine items, as they need to originate at an authorized producer.

Exploiting the impact of counterfeits on event traces, we model counterfeits as anomalies. We consider the set of all possible links between companies in the supply chain, i.e. pattern length $n = 2$. We then divide this set into “allowed” and “illegal” links. In order to detect a counterfeit the correlation agent S sends all patterns corresponding to “illegal” links (anomalies) to all event sources. If a match is detected, an investigation for counterfeits starts.

Note that our privacy guarantee only supports anomaly detection, i.e. detection of unwanted events, and not specification-based algorithms, i.e. detection of wanted behavior, since it reveals additional information in case of a match. This leakage is acceptable for anomalies, but in most cases not for compliant actions.

4 Privacy-Preserving Pattern Matching

4.1 Pattern Matching Function

An event trace t consists of a variable number of events e_j ($j = 1, \dots, m$). These events may be distributed across up to l parties X_k , i.e. each party X_k has a (usually but not necessarily consecutive) subset of the events e_j .

For now, we assume that each party is aware of the numbering of its events, i.e. each party knows j of its events e_j . We will revisit this assumption in Section 4.5 and show a method how to obtain the numbering without additional privacy-preserving computation.

Each event e_j has an event type $e_j.y$ from a finite set of types. A pattern p is a sequence of n event types $p.y_i$ ($i = 1, \dots, n$). It matches an event trace t , if there are n consecutive events e_j , such that $e_j.y = p.y_i$ ($j = \beta, \dots, \beta + n - 1, i = 1, \dots, n$). Loosely speaking, we slide the pattern over the event trace and if there is any position where event trace (completely) matches the pattern, the pattern matches the event trace.

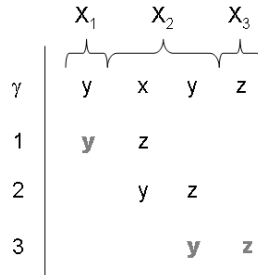


Fig. 2. Example of applying the basic mechanism

Consider the example of Figure 2. There are event types x, y, z and the event trace is y, x, y, z . It is distributed over the parties X_1, X_2, X_3 . The pattern p is y, z . The pattern p is slid across the event trace and is depicted for positions $\gamma = 1, 2, 3$. A bold, gray font indicates a match.

4.2 Protocol

We show how to implement this pattern matching algorithm using a distributed privacy-preserving protocol. Our privacy goal is that no information about the events (i.e. their type) is revealed, if the pattern does not match. If the pattern matches, the sources of the events may be revealed. Their type is implicitly revealed by the match. Patterns are public and may be revealed to the data sources.

We use a pseudo-random function $PRF(\cdot, key)$ with a key key as the basis of our scheme. The secret key is known to all event sources X_k , but not the central correlation agent S . We assume that the output of the pseudo-random function cannot be distinguished from truly random numbers, i.e. given pairs $m_i, PRF(m_i, key)$ and a number r it is impossible to determine whether $r = PRF(m, key)$ for any $m \neq m_i$. In practice one uses a message authentication code (MAC) function which is resistant to MAC forgery.

Let n be the length of the pattern. The correlation agent S sends the pattern p to each event source X_k . Each party X_k looks up its events e_j . Let \mathbb{J}_k be the set of numbers of found events at X_k .

X_k now considers each possible combination of positions $\gamma \in \{\min(j|j \in \mathbb{J}_k) - n + 1, \dots, \max(j|j \in \mathbb{J}_k)\}$ in its events and corresponding positions $\delta \in \{i|0 \leq i < n \wedge \gamma + i \in \mathbb{J}_k\}$ in the pattern. For each pair γ, δ it computes a hashed value $x_{\gamma, \delta}$. The value $x_{\gamma, \delta}$ is the keyed hash of the concatenation of γ and δ , if there is a match of event and pattern at this pair of positions. Since the correlation agent S does not know the PRF key key , this maintains the privacy of the match towards it.

The key insight is that, since the PRF key key is known to all event sources, the hashes of other sources are known in case of a match. We therefore assign a special role to the event source matching the last pattern position ($\delta = n - 1$). It does not compute the keyed hash in case of a match, but computes the hashes of all other pairs (which may be at other sources) and then their negated sum, such that the subset of $x_{\gamma, \delta}$ for this pattern position in the event trace adds up to 0.

In case of mismatch, X_k chooses an uniform random number r from the domain of the pseudo-random function. If any source for a pattern position chooses a random number (i.e. there is a mismatch), the sum will be random as well.

The formula for $x_{\gamma, \delta}$ is

$$x_{\gamma, \delta} = \begin{cases} PRF(\gamma, \delta, key) & \text{if } e_{\gamma+\delta} \cdot y = p \cdot y_{\delta+1} \wedge \delta < n - 1 \\ - \sum_{i=0}^{n-2} PRF(\gamma, i, key) & \text{if } e_{\gamma+\delta} \cdot y = p \cdot y_{\delta+1} \wedge \delta = n - 1 \\ r & \text{if } e_{\gamma+\delta} \cdot y \neq p \cdot y_{\delta+1} \end{cases}$$

In order to not reveal information about the position of a match X_k permutes its set of $x_{\gamma, \delta}$. But, since there are n $x_{\gamma, \delta}$ for each γ with $\delta = 0, \dots, n - 1$, X_k may reveal the δ for each $x_{\gamma, \delta}$. So, X_k sends to S a randomly permuted set of pairs $\langle x_{\gamma, \delta}, \delta \rangle$.

Note that, if events at one party are consecutive, there are $n|\mathbb{J}_k|$ such pairs, i.e. we reveal the number of events an event source has. To conceal that an event source has no events it can send $r' \cdot n$ pairs with a random number for $x_{\gamma, \delta}$. To conceal the number of events all parties must agree on an upper bound u and always send $u \cdot n$ pairs (padded with random numbers).

After receiving the pairs the correlation agent S sorts all of them in ascending order of their second value δ in the pair. S computes the sum for each possible combination τ_1, \dots, τ_n of pairs $\langle x_{\tau_1}, 0 \rangle, \dots, \langle x_{\tau_n}, n - 1 \rangle$ that spans all values of δ . Due to the algorithm for computing $x_{\gamma, \delta}$ this

$\sum_{i=1}^n x_{\tau_i} = 0$ will be equal to 0, if the pattern p matches the event trace t .

Consider party X_2 in the example from Figure 2: it has events at positions 2 and 3 and needs to compare for pattern positions $\gamma = 1, 2, 3$. It computes the pairs $\langle x_{1,1} = r, 1 \rangle$, $\langle x_{2,0} = r', 0 \rangle$, $\langle x_{2,1} = r'', 1 \rangle$ and $\langle x_{3,0} = PRF(3.0, key), 0 \rangle$. Since the length of the pattern is equal or below the number of its consecutive events and a match at positions inside its events can be determined by itself, but did not occur, X_2 can choose to omit the pairs for $\gamma = 2$. Party X_3 has an event at position 4 and computes $\langle x_{3,1} = -PRF(3.0, key), 1 \rangle$. It holds that $x_{3,0} + x_{3,1} = 0$, such that the correlation agent can detect the match.

4.3 Determining the Bit-Length of the PRF

Let κ be the bit-length of the PRF used above. We need to determine κ , such that false positives are unlikely. The correlation agent S receives nlu pairs using the algorithm above. It can then form $(lu)^n$ possible combinations of pairs, such that the following must hold

$$(lu)^n \ll 2^\kappa$$

for false positives to be negligible.

In our use case example given in Section 5 we have $l = 15$, $u = 1$ and $n = 2$, i.e. $\kappa \gg 7.81$. In this case we can save communication cost and even condense common PRF functions, such as HMAC based on SHA-1, by sending only the first 32 bits.

4.4 Security

As the event sources X_k do not share data with each other, but only with the correlation agent S , there is no risk of revealing information to other sources. We only need to prove privacy towards S . Assume the simplest attack where S tries to infer the event type of some victim X_\star . We argue that by using our algorithm it cannot do so and does not obtain any additional knowledge about the events at X_\star , i.e. our algorithm preserves the privacy of the events. We play the following game: S sends some pattern p of his choice to all X_k and receives the pairs $\langle x_{\gamma,\delta}, \delta \rangle$ in return including $\langle x_{\gamma^\star,\delta^\star}, \delta^\star \rangle$ from X_\star . We assume that the pattern p does not match the event trace beginning at position γ^\star , since that would reveal X_\star 's type from the result of the comparison. S is then asked to tell whether $x_{\gamma^\star,\delta^\star}$ is the PRF or a random number, i.e. whether X_\star has a matching event type at pattern position δ^\star .

Theorem 1 *Let m be the maximum number of events. If any adversary S wins the game above with probability $\frac{1}{2} + \epsilon$, then there is an algorithm \mathcal{B} that successfully distinguishes PRF outputs with probability at least $\frac{\epsilon}{m}$.*

Proof. If S outputs random number, \mathcal{B} outputs a random guess for the PRF pair. If S outputs PRF, \mathcal{B} chooses a random starting position j ($j \in \{0, \dots, m - 1\}$) for γ^* . S knows δ^* of x_{γ^*, δ^*} which is presumably $PRF(\gamma^*. \delta^*, key)$. So \mathcal{B} outputs $j. \delta^*$, x_{γ^*, δ^*} as its guess for the PRF pair. Its chances of success are $\frac{\epsilon}{m}$ (which is independent of the bit-length κ of the PRF).

Our security model could be translated into the semi-honest model for SMC [8]. Yet the ideal functionality is difficult to specify. Each event source's input are its events and their numbers. The correlation agent has no input, but the pattern is part of the (public) function to be computed. The output is whether there is a match and in case of a match the positions in the pattern of the sources' events comprising the match.

On the one hand, this is limited compared to other possible ideal SMC functionalities. E.g., SMC would allow implementing a function where the event sources input their events and correlation agent the pattern. The output would be a bit whether there is a match or not. This would clearly improve security by privacy for the pattern and privacy in the case of a match, but we remind the reader that we chose the function, such that its implementation can be efficient.

On the other hand, our security definition extends semi-honest security. No matter how the correlation agent behaves it is not able to infer information. This does not yet correspond to malicious or covert adversaries, since we do not protect the integrity of the computation, but confidentiality holds even against active adversaries.

4.5 Detection of Missing Events

So far we made the assumption that each party X_k is aware of the numbering of its events, i.e. each party knows the j of its events e_j . Since we are using RFID tags to generate the events, the simplest method to achieve this is to store j on the RFID tag. After reading the tag and storing the event in its database X_k updates j to $j + 1$ on the tag.

In order to raise the bar for a counterfeiter, the initial number should be randomized. Nevertheless a counterfeiter may interfere with the supply chain and alter the stored number of counterfeit goods. This may create overlapping event numbers which the pattern matching algorithm will still detect or missing event numbers which require additional consideration.

We extend our pattern matching algorithm to be able to detect missing events. The correlation agent sends a pattern p with event type $p.y_i = \star$ which matches an event trace t at position β , if no event source X_k has any event $e_{\beta+i-1}$.

The idea is very similar to the one for pattern matching. If no event source has an event, everyone knows the hashes of the other sources, such that we can control the sum. The difference is that this time the sum is computed for a specific pattern position $\delta = i$, such that this sum must be a summand $x_{\gamma,i} = PRF(\gamma.i, key)$ for the sum computed as above.

We assign the special role to event source X_l . Each party X_k ($1 \leq k < l$), i.e. everybody except X_l computes

$$x_{\gamma,i,k}^* = \begin{cases} PRF(\gamma.i.k, key) & \text{if } \gamma + i \notin \mathbb{J}_k \\ r & \text{if } \gamma + i \in \mathbb{J}_k \end{cases}$$

X_l computes

$$x_{\gamma,i,l}^* = \begin{cases} -\sum_{k=1}^{l-1} PRF(\gamma.i.k, key) + PRF(\gamma.i, key) & \text{if } \gamma + i \notin \mathbb{J}_k \\ r & \text{if } \gamma + i \in \mathbb{J}_k \end{cases}$$

The detection algorithm at the correlation agent S is slightly different and actually has become simpler. Note that S knows i where $p.y_i = \star$. It computes the sum $x_{\gamma,i} = \sum_{k=1}^l x_{\gamma,i,k}^*$ over all event sources and uses this one value $x_{\gamma,i}$ in the above detection algorithm (where there used to be l).

We omit the security proof for brevity, since it follows the same construction as before. Simply note that detection of a missing event without a complete pattern match implies PRF distinguishability.

5 Evaluation

We evaluate the performance and communication cost of our algorithm as used for anti-counterfeiting. We model the supply chain as a q stage process. The length m of a trace is then equal to q .

First, we estimate the number of necessary patterns (“illegal links”). Let $f(\chi)$ be the discrete probability density function of the number of companies in one stage of the supply chain and $F(\chi)$ be its cumulative distribution function. We assume all stages are independent identically distributed. Then the expected number of allowed links between two stages is

$$a = E(\chi^2) = E(\chi)E(\chi)$$

The expected number of all links is $qE(\chi)(qE(\chi) - 1)$ and the expected number of illegal links is

$$b = qE(\chi)(qE(\chi) - 1) - a(q - 1)$$

Thus we expect b patterns in the system.

Second, we estimate the necessary number of event traces that cover all “allowed” links called the clean set, since it may not contain any counterfeits. Between each two consecutive stages there are a allowed links and each needs to be present in at least one trace. We assume that the probability of links occurring at different stages is independent. We observe q stages in parallel and the number of necessary event traces is determined by the maximum number of links between any two stages. The probability density function of the maximum is given by

$$g(\chi^2) = \sum_{i=1}^{q-1} \binom{q-1}{i} f(\chi^2)^i F(\chi^2 - 1)^{q-1-i}$$

The expected value of the maximum is

$$c = \sum_{\chi^2} \chi^2 g(\chi^2)$$

If we assume that the probability of the occurrence of each link is uniform, then determining the expected number of traces, such that each link occurs at least once is an instance of the coupon collector’s problem. The expected number of traces necessary can be then computed as

$$d = \lceil c \rceil \sum_{i=1}^{\lceil c \rceil} \frac{1}{i}$$

The formula for computing the expected number of traces necessary in case of a non-uniform distribution can be found in [7].

We will continue using numbers inspired by a real-world example. Assume a supply chain with $q = 5$ stages and let the number of parties at a stage be uniformly distributed between 1 and 5. The expected number of parties at a stage $E(\chi) = 3$ and the expected number of parties in the entire supply chain is $l = 15$.

Exceptional situations may significantly increase the necessary size of the clean set, such that in practice one can expect some false positives due to cases, such as a return delivery, and read errors, such as a failing RFID tag.

The expected number of allowed links between two stages is $a = 9$. The expected number of illegal links, i.e. patterns is $b = 174$. The expected value of the maximum number of links between any two stages is $c = 16.1$. The expected number of traces necessary in the clean set is then $d = 58.5$.

An inherent problem with our algorithm is that the detection of an anomaly is exponential in the length of the pattern. The algorithm has to exhaustively search $O((lu)^n)$ possible combinations. In our use case of counterfeit detection in supply chains this does not pose a problem, since $n = 2$ and the search algorithm can be further optimized using hash tables to $O(lu)$ expected time complexity.

We implemented the detection algorithm using 160-bit HMAC based SHA-1 on a 2.4 GHz Intel Xeon machine using Java 1.5 on Windows XP. We generated random patterns of a given length and matched them to randomly generated strings. We report the average of the spent wall clock time of 1000 runs of the matching algorithm. Figure 3 shows the results in milliseconds for a pattern length of 2 to 8 on a logarithmic scale. Even this non-optimized algorithm can perform counterfeit detection ($n = 2$) for a single pattern in less than 2 ms. Furthermore we see that detection times for pattern lengths up to 5 seem acceptable (< 1 s), but this assessment obviously depends on the rate of incoming events.

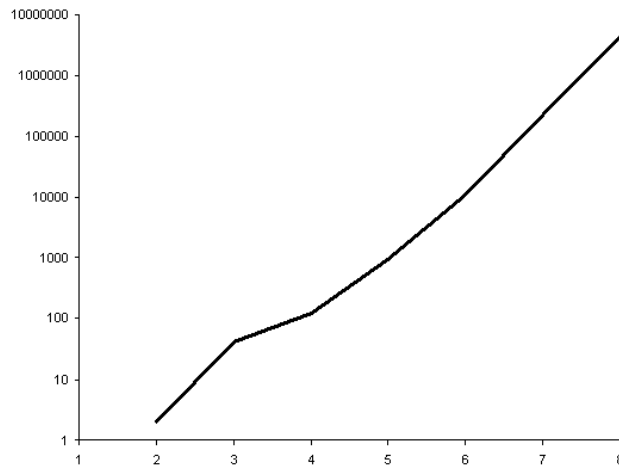


Fig. 3. Running time in ms over pattern length

Computation of the detection input at the event sources scales linearly in the pattern length $O(n|\mathbb{J}_k|)$. On the test hardware we computed 1000 HMACs in 73 milliseconds and therefore this is not expected to be a performance bottleneck.

For each tag we need to communicate one PRF value per event in the pattern (n), per event source (l), per match up to the limit (u) and per pattern to match (b). The communication complexity is consequently $O(nlub)$ per tag. Assuming the PRF length of 32 bits (4 bytes) from Section 4.3 we calculate 20 KBytes for our example.

We can estimate the communication cost when using secure multi-party computation with [2]. For this method we need a circuit implementing the pattern matching consisting of gates implementing any binary function. We construct this circuit from individual components from a library we have developed over a course of projects.

We start with a circuit that first sorts the events using a sorting network. Note that our algorithm does that implicitly. We continue our example and assume $l = 15$ parties which supply $u = 1$ event each. The event type corresponds to the party identifier and consists of 4 bits. The numbering of events consists of 8 bits. A sort-and-exchange operation in a sorting network can then be implemented using 73 gates. Using Batcher’s construction we need 80 such operations. The result must be compared to $b = 174$ at $l * u - n + 1 = 14$ positions with $n = 2$ events. The results of this comparison are condensed into the output bit – match or no match.

The entire circuit consists of 47252 gates. For each gate we need $4l = 60$ keys of 80 bits which results in 216 MByte which needs to be communicated by each computing party. This results in an overall communication of 3.2 GByte – compared to our 20 KByte a factor of more than $\cdot 10^6$. Our exponential computation complexity seems very reasonable compared with these absolute numbers.

6 Conclusion

We presented a privacy-preserving pattern matching algorithm. This algorithm is provably secure, but with limited privacy protection by its ideal functionality. We show how the matching algorithm and the privacy guarantee can be used to implement distributed anomaly detection. We show that computation and communication costs are acceptable for counterfeit detection in an example supply chain.

Our work extends the state-of-the-art with a novel protocol for practical privacy-preserving distributed event correlation. Therefore new de-

tection algorithms, such as our anomaly detection, can be implemented with privacy for the event sources.

We differ from previous approaches by implementing provable security. We counterbalance the resulting performance penalty by limiting the privacy guaranteed by the ideal functionality. We anticipate that this strategy may lead to further practical and privacy-preserving protocols for different correlation algorithms.

7 Acknowledgements

The developments presented in this paper were partly funded by the European Commission through the ICT program under Framework 7 grant 213531 to the *SecureSCM* project and by the German BMBF through the grant to the *Polytos* project.

References

1. B. Applebaum, M. Freedman, H. Ringberg, M. Caesar, and J. Rexford. Collaborative, Privacy-Preserving Data Aggregation At Scale. Available at <http://eprint.iacr.org/2009/180.pdf>, 2009.
2. A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP: a system for secure multi-party computation. *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, 2008.
3. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proceedings of the 20th annual ACM symposium on Theory of computing*, 1988.
4. M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Security through Private Information Aggregation. Available at <http://arxiv1.library.cornell.edu/abs/0903.4258>, 2009.
5. W. Du, and M. Atallah. Secure Multi-Party Computation Problems and Their Applications: A Review and Open Problems. *Proceedings of the Workshop on New Security Paradigms*, 2001.
6. European Commission. Report on Community Customs Activities on Counterfeit and Piracy. 2008.
7. P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics* 39(3), 1992.
8. O. Goldreich. Secure Multi-party Computation. Available at www.wisdom.weizmann.ac.il/~oded/pp.html, 2002.
9. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987.
10. C. Hazay, and Y. Lindell. Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. *Proceedings of the 5th Theory of Cryptography Conference*, 2008.
11. F. Kerschbaum. Practical Privacy-Preserving Benchmarking. *Proceedings of the 23rd IFIP International Information Security Conference*, 2008.

12. F. Kerschbaum, D. Dahlmeier, A. Schröpfer, and D. Biswas. On the Practical Importance of Communication Complexity for Secure Multi-Party Computation Protocols. *Proceedings of the 24th ACM Symposium on Applied Computing*, 2009.
13. C. Kuerschner, F. Thiesse, and E. Fleisch. An analysis of data-on-tag concepts in manufacturing. *Proceedings of the 3rd Konferenz Ubiquitäre und Mobile Informationssysteme*, 2008.
14. A. Lee, P. Tabriz, and N. Borisov. A Privacy-Preserving Interdomain Audit Framework. *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, 2006.
15. M. Lehtonen, F. Michahelles, and E. Fleisch. How to Detect Cloned Tags in a Reliable Way from Incomplete RFID Traces. *Proceedings of the IEEE RFID Conference*, 2009.
16. P. Lincoln, P. Porras, and V. Shmatikov. Privacy-Preserving Sharing and Correlation of Security Alerts. *Proceedings of the USENIX Security Symposium*, 2004.
17. M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo. Towards Collaborative Security and P2P Intrusion Detection. *Proceedings of the Information Assurance Workshop*, 2005.
18. L. Mirowski, and J. Hartnett. Deckard: A System to Detect Change of RFID Tag Ownership. *International Journal of Computer Science and Network Security* 7(7), 2007.
19. N. Oertel. Tracking based product authentication: Catching intruders in the supply chain. *Proceedings of the 17th European Conference on Information Systems*, 2008.
20. J. Parekh, K. Wang, and S. Stolfo. Privacy-Preserving Payload-Based Correlation for Accurate Malicious Traffic Detection. *Proceedings of the SIGCOMM Workshop on Large-Scale Attack Defense*, 2006.
21. P. Porras, and V. Shmatikov. Large-Scale Collection and Sanitization of Network Security Data: Risks and Challenges. *Proceedings of the Workshop on New Security Paradigms*, 2006.
22. B. Santos, and L. Smith. RFID in the Supply Chain: Panacea or Pandora's Box? *Communications of the ACM* 51(10), 2008.
23. S. Sarma, D. Brock, and D. Engels. Radio frequency identification and the electronic product code. *IEEE Micro* 21(6), 2001.
24. T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC Network – The Potential of RFID in Anti-Counterfeiting. *Proceedings of the 20th ACM Symposium on Applied Computing*, 2005.
25. A. Yao. Protocols for Secure Computations. *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1982.
26. D. Zanetti, L. Fellmann, and S. Capkun. Privacy-preserving Clone Detection for RFID-enabled Supply Chains. *Proceedings of the IEEE International Conference on RFID*, 2010.