

Privacy-Preserving Computation of Benchmarks on Item-Level Data Using RFID

Florian Kerschbaum, Nina Oertel, Leonardo Weiss Ferreira Chaves

SAP Research

Karlsruhe, Germany

{florian.kerschbaum | nina.oertel | leonardo.weiss.f.chaves}@sap.com

ABSTRACT

Currently, companies are about to optimize their internal processes by monitoring items they handle with Radio Frequency Identification (RFID). However, there is a risk that sensitive information is disclosed when sharing RFID data with other companies. Therefore, companies are unwilling to share RFID data. At first glance, Secure Multi-Party Computation (SMC) might reconcile data sharing with the privacy concerns. However, SMC requires the collaboration of all parties involved in a protocol. This prevents using SMC for many applications based on item-level RFID data collected in supply chains, since some parties may be competitors or have conflicting interests. We present protocols for securely and privately computing item-level metrics using only existing communication links (e.g., messages stored on RFID tags) and an oblivious third party. This enables optimizing the supply chain using novel item-level metrics without compromising sensitive information.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*distributed applications*; D.4.6 [Operating Systems]: Security and Protection—*Cryptographic controls*

General Terms

Algorithms, Security

Keywords

RFID, Supply Chain Metrics, Secure Multi-Party Computation

1. INTRODUCTION

Attaching RFID tags to items in a supply chain allows them to be tracked as they are handled by the individual supply chain partners [20]. Each partner collects the processing time and item identifier in each production step.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WiSec'10, March 22–24, 2010, Hoboken, New Jersey, USA.

Copyright 2010 ACM 978-1-60558-923-7/10/03 ...\$10.00.

Aggregating this information along the supply chain allows the supply chain partners to compute metrics (also called key performance indicators) to measure the performance of the entire supply chain. Such metrics can be used to improve the performance of the supply chain, e.g., by comparing (benchmarking) different supply chains or different paths of one supply chain.

However, companies are generally reluctant to share fine-granular RFID event data with business partners [13, 19]. A risk associated with sharing RFID event data is the reconstruction of strategic relationships with other companies, suppliers, customers and the design of the distribution channels. Business secrets about future strategic products or process restructurings might be revealed. Thus companies are looking for ways of reaping the benefits of RFID in the supply chain, but at the same time sharing as little information as possible to mitigate the associated risks.

Consider a three-staged supply chain. In the first stage two suppliers produce interchangeable goods. They sell their goods to a supplier on the second stage which uses the goods to make another good. This is sold to the producer on the third stage which finally sells it to customers. This producer is interested in measuring the supply chain performance and identifying potential bottlenecks to reduce manufacturing costs. A suitable metric is the aggregate time a good (part or product) spent in any warehouse along the supply chain. The duration of stay in one's warehouse can be considered sensitive and private information to each supply chain partner. The information which and how many parts have been bought from which supplier in the first stage can be considered private to the supplier in the second stage.

Secure Multi-Party Computation (SMC) [2, 8, 21] offers a cryptographic solution where multiple parties can compute a joint function without revealing the inputs. In the aforementioned supply chain scenario, traditional SMC could compute the sum of the time a good spends in all warehouses. This would ensure the required confidentiality of the warehouse stay times. However, if the sum is calculated only among the owners of an item, sensitive information is revealed: If the supply chain partners in all stages calculate the sum using SMC, the producer will learn which item contains parts from which supplier in the first stage. This reveals operational information about the supplier in the second stage and potentially his cost calculation to the producer. Consequently, the producer could demand cheaper goods from the supplier in the second stage. If the summation is performed among all supply chain partners, regardless of whether they possessed an item, each supplier in the first stage will at

least learn the number of items the other supplier in the first stage has sold. This is also true for a secure computation over all items, since the length of the input is not hidden. This information is particularly sensitive, since the suppliers in the first stage are competitors for the same good and are selling to the same buyer.

1.1 RFID Metrics

Metrics are a tool for decision-makers to identify current shortcomings and value-creating improvements in supply chain execution. RFID metrics are usually more fine-grained than traditional supply chain metrics, thus leading to an improved decision quality [9].

We model the supply chain as an n -stage process with production facilities and warehouses. The customer is modeled as stage $n + 1$. The supply chain is divided into a manufacturing chain and a distribution chain.

In the manufacturing chain an item passes through each stage undergoing a manufacturing process. An output item at stage i requires a fixed number of input items at stage $i - 1$. Due to the RFID technology we can track each individual intermediate item built into a finished item. Let Ω_j be the set of intermediate items built into item j .

In the distribution chain items are distributed to the final customers and there are no more production facilities (only warehouses). Let η be the first stage of the distribution chain. We can track each individual item in the distribution chain using its unique RFID tag.

Let $t_{i,j}$ be the time spent by a particular item j in the warehouse at stage i . Let $u_{i,j}$ be the time spent by a particular item j in the production facility at stage i . In the distribution chain $u_{i,j} = 0$. Let $v_{i,j}$ be the time spent by a particular item j on the transport from stage i to stage $i + 1$. The first type of metric made possible by RFID technology are the aggregate times across the entire supply chain, i.e., manufacturing, storage and transportation time [4]. In the distribution chain we can compute the aggregate g_j of the times spent at the different stages:

$$g_j = \sum_{i=\eta}^n t_{i,j}$$

The definitions for u_j and v_j are analogous, but we focus on warehouse stay time in the following.

In the manufacturing chain several items may be combined to form the final item. Therefore we need an iterative definition of the warehousing time h_j . If item j is produced at stage i , then the manufacturing warehousing time h_j consists of the time spent in the warehouse in this stage ($t_{i,j}$), and the maximum time one of its parts or components spent in a warehouse ($\max_{k \in \Omega_j} h_k$):

$$h_j = t_{i,j} + \max_{k \in \Omega_j} h_k$$

The overall warehousing time t_j of an item j is

$$t_j = g_j + h_j$$

Ultimately we are interested in statistics about multiple items. Let θ be the number of finished items. Then the average warehousing time $avg(t)$ and the maximum warehousing time $max(t)$ are computed as follows.

$$\begin{aligned} avg(t) &= \frac{1}{\theta} \sum_j t_j \\ max(t) &= \max_j t_j \end{aligned}$$

The second type of metric made possible by RFID technology are fractional metrics which identify the fraction of all items that have a certain property, e.g., the fraction of returned items per supplier [4]. Let $\Phi_{l,m}$ be the set the items received by a particular supply chain partner X_m from supply chain partner X_l . Let Ψ be the set of returned (defective) items.

In the distribution chain the computation of the fractional metric, i.e. the number of returned defective items per supplier, f_l at supply chain partner X_l is simple again:

$$f_l = \frac{|\Phi_{l,m} \cap \Psi|}{|\Phi_{l,m}|}$$

In the manufacturing chain we need an iterative definition again. Let Ψ_i be the set of intermediate items at stage i that end up being defective (i.e., in Ψ)

$$\Psi_i = \{k | k \in \Omega_j \wedge j \in \Psi_{i+1}\}$$

Note that $\Psi_{\eta-1} = \Psi$.

For a supply chain partner X_l at stage i we then have

$$f_l = \frac{|\Phi_{l,m} \cap \Psi_i|}{|\Phi_{l,m}|}$$

1.2 Security Requirements

After describing the problem and the metrics to be computed, we specify the security requirements for our solution. We specify two security requirements: one on the definition of the function to be computed (supply chain visibility) and one on the protocols implementing the function (semi-honest security). For brevity we refer the reader to [7] for a definition of semi-honest security.

We introduce a semi-trusted third party T . This third party is supposed to learn neither the RFID event data (input) nor the resulting benchmarking metrics (output). As such it is not the trusted third party in the ideal model of SMC, i.e., it remains entirely oblivious to the computation. Nevertheless it is trusted not to collude with any of the supply chain partners. A good candidate for the semi-trusted third party would be a trade organization or standards body.

1.2.1 Supply Chain Visibility

An RFID event is a triple $\tau = \langle time, company, item \rangle$. A triple τ is valid, if it corresponds to a real supply chain delivery, e.g., it has been collected by reading the RFID tag. We assign a probability $p_i(\tau)$ that a supply chain partner X_i believes τ is a valid event. Supply chain visibility means that a supply chain partner can distinguish valid from invalid triples with high probability.

Our goal is to not increase supply chain visibility, i.e., partners should not learn to distinguish additional triples. Every partner X_i has some previous knowledge about supply chain events, simply from observing its operations. We therefore define an *a priori* probability $\bar{p}_i(\tau)$ for partner X_i to believe τ is valid before executing our protocols, but assuming he already knows the metrics that will be computed, and an *a posteriori* probability $\hat{p}_i(\tau)$ after executing our protocols.

DEFINITION 1. *We say our protocols are supply chain-private, if for all supply chain partners X_i and all triples τ the difference of a priori and a posteriori probability is a negligible function of the security parameter k :*

$$|\bar{p}_i(\tau) - \hat{p}_i(\tau)| < \frac{1}{\text{poly}(k)}$$

As described above, we model the supply chain as an n -stage process. We assume that each supply chain partner X_i at least knows the stage of the supply chain he is on. An important sub-requirement of supply chain privacy is that no supply chain partner learns the other partners on the same stage, in particular if they are supplying the same product. This prevents suppliers from excluding or spying on competition within their own stage and preserves the buyer's competitive advantage of independently selecting suppliers and customers.

2. RFID BENCHMARKING PROTOCOLS

2.1 Building Blocks

2.1.1 Homomorphic Encryption

In homomorphic encryption one operation on the ciphertexts produces an encryption of the result of a homomorphic operation on the plaintexts. In particular, we require the homomorphic operation to be addition (modulo a key-dependent constant). We experimented with two different encryption systems [15, 18] in the implementation. Let $E_X(x)$ denote the encryption of x with X 's public key and $D_X()$ the corresponding decryption with X 's private key, then Paillier's encryption system has the following property:

$$D_X(E_X(x) \cdot E_X(y)) = x + y$$

With simple arithmetic the following property can be derived

$$D_X(E_X(x)^y) = x \cdot y$$

2.1.2 Oblivious Transfer

As another building block our protocol uses Oblivious Transfer (OT). In 1-out-of-2 OT the sender has two secrets x_0 and x_1 and the receiver has one bit b . After the execution of the OT protocol, the receiver has obtained x_b , but has learnt nothing about x_{-b} , and the sender has not learnt b . The fastest known implementation of OT which is also used in our implementation is described in [17]. The efficient version of the protocol was proven secure under the (computational and decisional) Diffie-Hellman and the random oracle model assumptions. The authors also give a less efficient construction that is secure under the decisional Diffie-Hellman assumption alone (without random oracles) which we did not implement.

2.1.3 Secure Comparison

We also use secure protocols to compare two values. Since this problem was introduced by Yao [21], it is also called Yao's millionaires' problem. Imagine two millionaires that want to compare their wealth, but no one wants to reveal to the other his exact wealth. They can do so using a secure comparison protocol.

Let a be Alice's input and b be Bob's. Then a secure comparison protocol computes the bit $a < b$. This bit is learned by one party first which then may send it to the other party.

A variation of the secure comparison protocol is the split secure comparison protocol. Instead of one party learning

the result, both parties share the result in a split fashion, such that no party knows the result by itself. Let \oplus denote the exclusive-or operation. Then a split secure comparison protocol computes $c_A \oplus c_B = (a < b)$, such that Alice obtains only c_A and Bob only c_B . The general solution in [21] can be easily adapted to yield a split secure comparison protocol.

2.2 Aggregate Time Metrics

2.2.1 Simple Protocol

Let $X_{i_1} \rightarrow X_{i_2} \rightarrow \dots \rightarrow X_{i_n}$ be the path of item j in the supply chain. We compute aggregate manufacturing, storage and transportation time in parallel.

There is a logical triple t_j, u_j, v_j corresponding to each item j . Each time information is split into two parts, e.g., t_1 and t_2 . In most cases we omit the index j for readability, since we only need to consider shares for one item at a time. If we need to distinguish shares from two items j_1 and j_2 we write t_{1,j_1} , t_{1,j_2} and so forth. The second share t_2 is encrypted in a homomorphic encryption system $E_T()$ whose key is only known to the semi-trusted third party. Let \mathbb{Z}_q be the group of the homomorphic operation. It holds that

$$t_1 + t_2 \equiv t_j \pmod{q}$$

The information $t_1, E_T(t_2)$ can be either stored on the item or sent over the network when the item is shipped (along with the other time information). The system is initialized as follows: X_{i_1} reads the current time δ and stores $E_T(-\delta)$ for the item j . He chooses three random numbers r_1, r_2 and r_3 and stores $t_j = \langle r_1, E_T(-r_1) \rangle$, $u_j = \langle r_2, E_T(-r_2) \rangle$, $v_j = \langle r_3, E_T(-r_3) \rangle$ for the item j .

When an event for item j occurs at X_i , X_i reads the current time δ' . Then he chooses a random number r and computes $\alpha = E_T(\delta' - r) \cdot E_T(-\delta)$. He picks the corresponding time information depending on the type of the event. Without loss of generality, let the warehousing time be $t_j = \langle t_1, E_T(t_2) \rangle$. He finally computes the new warehousing time $t'_j = \langle t_1 + r, \alpha \cdot E_T(t_2) \rangle = \langle t_1 + r, E_T(t_2 + (\delta' - \delta) - r) \rangle$ and stores the new time information $E_T(-\delta')$ for the item j .

2.2.2 Security Analysis

Supply chain partner X_i receives the ciphertext of a time-stamp $E_T(-\delta)$ and three tuples of type $\langle r, E_T(x_j - r) \rangle$. This can be simulated in the semi-honest model by four random ciphertexts, since X_i cannot decrypt $E_T()$, and three random numbers, since the plaintexts have been chosen randomly.

2.2.3 Extension in the Manufacturing Chain

If X_i is part of the manufacturing chain he might need to compute the maximum of two or more received intermediate items j_1 and j_2 . E.g., for warehousing, he receives two pieces of time information $t_{j_1} = \langle t_{1,j_1}, E_T(t_{2,j_1}) \rangle$ and $t_{j_2} = \langle t_{1,j_2}, E_T(t_{2,j_2}) \rangle$. It holds that

$$t_{j_1} < t_{j_2} \iff t_{1,j_1} - t_{1,j_2} < t_{2,j_2} - t_{2,j_1}$$

X_i computes $\beta = E_T(t_{2,j_1})^{-1} \cdot E_T(t_{2,j_2})$ and sends the result to the semi-trusted third party. In the simple case the semi-trusted third party decrypts β and they engage in a secure comparison protocol. X_i learns the result of the comparison (but withholds it from the semi-trusted third party) and chooses the received time information accordingly.

The learnt bit from the comparison reveals additional information to X_i . In a strict semi-honest model of the entire supply chain this is not admissible. X_i and the semi-trusted third party can therefore hide the result of the comparison. They engage in a split secure comparison and X_i obtains c_A and the semi-trusted third party c_B , such that $c_A \oplus c_B = (t_{j_1} < t_{j_2})$.

X_i flips a random coin and chooses one of the two pieces of time information as basis for the next computations. Without loss of generality, let t_{j_1} be the chosen information. X_i computes

$$\begin{aligned}\rho_0 &= E_T(t_{2,j_1}) \\ \rho_1 &= E_T(t_{2,j_2}) \cdot E_T(t_{1,j_2}) \cdot E_T(-t_{1,j_1}) \\ &= E_T(t_{2,j_2} + t_{1,j_2} - t_{1,j_1})\end{aligned}$$

He prepares two messages $\sigma_i = \rho_{i \oplus c_A}$ ($i \in \{0, 1\}$) as a sender in an OT. The semi-trusted third party acts as the receiver in the OT and chooses σ_{c_B} . It re-randomizes the received ciphertext by homomorphically adding “0” $\sigma'_{c_B} = \sigma_{c_B} \cdot E_T(0)$ and returns σ'_{c_B} to X_i . X_i sets as the new time information $t_j = t_{1,j_1}, \sigma'_{c_B}$. X_i can now add the local time $\delta' - \delta$ as in the above protocol.

As we compute manufacturing, storage and transportation time in parallel it is advantageous to compute the maximum of the sum of the three, instead of three separate maxima. If X_i computes three separate maxima, the overall lifetime of the item will not reflect the lifetime of any of its parts. If the sum of the three aggregate times is compared, the lifetime of an item is the maximum lifetime of its parts. Literature has not yet provided a conclusive answer which method of metric computation provides the best results.

In order to compute the maximum lifetime, X_i can use the protocol above with some minor modifications. First X_i computes the lifetime $s_j = \langle s_1, E_T(s_2) \rangle$ of each item j as

$$\begin{aligned}s_1 &= t_1 + u_1 + v_1 \\ E_T(s_2) &= E_T(t_2 + u_2 + v_2) = E_T(t_2) \cdot E_T(u_2) \cdot E_T(v_2)\end{aligned}$$

Then X_i and the semi-trusted third party engage in a secure comparison protocol for s_{j_1} and s_{j_2} . In the OT protocol X_i prepares the message ρ_0 as concatenations of $E_T(t_{2,j_1})$, $E_T(u_{2,j_1})$ and $E_T(v_{2,j_1})$. The message ρ_1 is prepared correspondingly as a concatenation of the adjusted encrypted time information shares. X_i and the semi-trusted third party engage in the OT protocol and X_i obtains the three time information.

If the item j is composed of more than two items, the computation of the maximum can be done sequentially. Since X_i does not learn which is the maximum time information after a comparison protocol with OT, X_i and the semi-trusted third party can engage in another comparison (with OT) with the just computed, intermediate maximum and the next element that composes j . They simply need to run the protocol once for each item (except for the last one).

2.2.4 Security Analysis

We analyze the fully semi-honest case. The semi-trusted third party first receives an encrypted random number, since the difference of two uniform random numbers is uniform. The result of the split secure comparison is randomly distributed. In the Oblivious Transfer, the semi-trusted third party receives another random number, due to the random choice of the basis by X_i . X_i receives a random ciphertext,

because the semi-trusted third party has re-randomized it and X_i cannot decrypt it. A simulator in the semi-honest model would simulate this with random numbers and random ciphertexts.

2.2.5 Computation of the Results

Once a number of items has reached X_n at stage n , X_n should compute the final metrics $avg(t)$ and $max(t)$. For $avg(t)$ he identifies the set Θ of items to use in benchmarking. He then computes $avg(t_1) = \sum_{j \in \Theta} t_{1,j}$ and $E_T(avg(t_2)) = \prod_{j \in \Theta} E_T(t_{2,j}) = E_T(\sum_{j \in \Theta} t_{2,j})$. He then sends $E_T(avg(t_2))$ to the semi-trusted third party and receives the plaintext $avg(t_2) = \sum_{j \in \Theta} t_{2,j}$ in return. The result is $avg(t) = avg(t_1) + avg(t_2)$. Care must be taken that the result does not overflow the group of the plaintext in the homomorphic encryption, but given current key sizes this does not seem to be a problem.

The computation of the maximum $max(t) = \max_{j \in \Theta} t_j$ is sequential. X_n picks the first element and sets it as the intermediate maximum. X_n and the semi-trusted third party engage in the same protocol as in Section 2.2.3 for each subsequent item.

In the simple case X_n will learn the result of the comparison, but this potentially reveals the entire order of the elements. In the fully semi-honest case X_n will not learn which maximum element was chosen, as above, but has the shared (encrypted and unencrypted part) version of it.

Let $max(t) = \langle max(t_1), E_T(max(t_2)) \rangle$ be the final maximum element. X_n sends $E_T(max(t_2))$ to the semi-trusted third party and receives the plaintext $max(t_2)$ in return.

2.2.6 Security Analysis

In the computation of $avg(t)$ the semi-trusted third party receives the ciphertext of a random number, since the encrypted parts are chosen randomly. In the computation of $max(t)$ X_n and the semi-trusted third party first engage in a number of protocols which are secure according to above analysis. Then the semi-trusted third party receives a ciphertext of a random number again. In both cases X_n learns the resulting metric which is allowed, since it is the output of the algorithm.

2.3 Fractional Metrics

2.3.1 Protocol

In a fractional metric one supply chain partner identifies the marked items Ψ , i.e., the returned items. Without loss of generality, let X_n be that partner and he wants to communicate with X_{n-1} .

Note that X_n can compute f_{n-1} locally and we can turn to the problem of computing f_{n-2} . For each item j in Ψ X_n computes an encrypted bit: $b_j = E_T(1)$ if the item is marked and $b_j = E_T(0)$ if it is not. X_n then sends each j in $\Phi_{n-1,n}$ and its encrypted bit b_j to X_{n-1} .

X_{n-1} receives all j from all X_n and divides them in sets for the corresponding partners in stage $n-2$. If X_{n-1} is part of the manufacturing chain, he looks up Ω_j for that. For each partner in stage $n-2$ he computes the input $x_{n-2} = \prod_j b_j = E_T(|\Phi_{n-2,n-1} \cap \Psi_{n-1}|)$. He sends x_{n-2} to the semi-trusted third party and receives the plaintext $|\Phi_{n-2,n-1} \cap \Psi_{n-1}|$ in return. The result is computed as $f_{n-2} = \frac{|\Phi_{n-2,n-1} \cap \Psi_{n-1}|}{|\Phi_{n-2,n-1}|}$.

The protocol can be iteratively continued until stage 1 is reached.

2.3.2 Security Analysis

X_{n-1} receives the set of items in $\Phi_{n-1,n}$ which he already knows and $|\Phi_{n-1,n}|$ ciphertexts. This can be simulated by $|\Phi_{n-1,n}|$ random ciphertexts.

The semi-trusted third party receives a random number, since it does not know $|\Phi_{n-1,n}|$. Strictly speaking it learns more information, since it learns the number of marked items. This could be easily prevented by X_{n-1} first homomorphically adding a random number.

X_{n-1} receives the output of the computation, i.e., number of marked items, since X_{n-1} knows $|\Phi_{n-1,n}|$.

2.4 Supply Chain Privacy

We need to show that our protocols are supply chain-private according to Definition 1. We have given simulators for the messages received from another supply chain partner in aggregate time metrics in Section 2.2.2 and in fractional metrics in Section 2.3.2. The output of one supply chain party are the messages received by another supply chain party at the next stage. So no party learns anything outside the metrics and the knowledge from their operations.

We can construct an algorithm \mathcal{A} that computes the differences $|\hat{p}_i(\tau) - \tilde{p}_i(\tau)|$ of a priori and a posteriori probabilities. The information obtained from our protocols by the metrics is already considered in the a priori probabilities. This leaves the messages received from the other supply chain partners as additional input provided by our protocols. Therefore if any difference is not negligible, algorithm \mathcal{A} could distinguish the view of a supply chain partner in the real protocol from the view our simulator in the proofs. The existence of algorithm \mathcal{A} then contradicts our proof of semi-honest security.

3. PERFORMANCE EVALUATION

We implemented the semi-honest scheme in Java 1.6. The Java language offers a standard-library package for big integer arithmetic which is the basis of our cryptography library. We implemented two versions of the algorithms: one using Paillier’s crypto system [18] and one using Naccache-Stern crypto system [15] for the homomorphic encryption system.

In our first implementation key length of Paillier’s crypto system was 1024 bits for an RSA modulus. For comparison we used the protocol from [10, 12] for its performance and its easy conversion to a split version. In order to compare a 1024 bit number we need to generate a 3072 bit RSA key and compute in the corresponding field. The OT protocol implemented is [17].

In our second implementation using the Naccache-Stern crypto system we used a key size of 2048 bits, since it is not clear how the small factors in Naccache-Stern affect factoring. We used a field size of 64 bits for the homomorphic operation, such that we can use a 1024 bit key for the comparison protocol from [10, 12]. The other implementation details remain unchanged.

We measured the runtime of the following protocols: aggregate time metric *Agg*, aggregate time metric in the manufacturing chain *AggMan*, and fractional metric *Frac*. In the *Agg* protocol, one time information is updated with the time difference to the stored time. This protocol is executed for every event and can be performed non-interactively at the supply chain partner. The *AggMan* is an interactive protocol between a supply chain partner and the semi-trusted

third party. We assumed that two items compose the next stage item. For c items the same protocol needs to be performed $c - 1$ times. The *Frac* protocol implements the protocol step that needs to be performed for every item in a fractional metric. One additional decryption system for all items needs to be performed. We did not measure this.

We evaluated the performance on a 2.4 GHz Xeon server under Linux 2.6. The results are summarized in Table 1.

	Paillier	Naccache-Stern
<i>Agg</i>	190 ms	10 ms
<i>AggMan</i>	2484 ms	1018 ms
<i>Frac</i>	71 ms	3 ms

Table 1: Performance results

The results clearly indicate that the implementation with the Naccache-Stern crypto system is faster. We attribute this to the fact that the particularly slow operation of the Naccache-Stern crypto system, decryption, only occurs once in the *AggMan* protocol. Other operations in the Naccache-Stern crypto system, such as encryption and homomorphic operations are very fast. We believe that the single decryption for computing fractional metrics is negligible compared to the number of items involved in the computation.

For computing aggregate time metrics it can be beneficial to store information on the tag. The encrypted timestamp and the encrypted time information must be forwarded along with the item and should be stored on the tag. The timestamp has the length of one ciphertext, i.e., 2048 bits (256 byte) in both of our implementations. One time information contains an element of the homomorphic group, i.e., 1024 bits (128 byte) in Paillier’s crypto system and 64 bits (8 byte) in the Naccache-Stern crypto system, and a ciphertext of again 2048 bits (256 byte). Assuming three time informations (warehousing, production and transportation), it sums to 1408 byte for Paillier’s crypto system and 1048 byte for the Naccache-Stern crypto system.

4. RELATED WORK

The original SMC protocols [2, 8] required full connectivity between all parties. Several protocols have been developed since that allow secure computation without full connectivity. Our protocols use the same approach by limiting the connectivity to the links of the supply chain.

The first approach is to separate computation servers and clients. Computation servers perform a standard SMC protocol, e.g., [2, 8, 21] and clients supply input. The idea is that clients divide their input according to a secret sharing scheme and distribute the shares among the computation servers. This idea was first introduced in [16] for two computation servers using a two party secure computation protocol [21]. Later this was extended to multiple computation servers in [5] and reduced to one server in [11]. The idea was widely adopted in SMC implementations [1, 3]. The collusion threshold is, of course, reduced to a fraction of the computation servers.

The second approach is called almost-everywhere secure computation [6] and still involves all parties in the computation, but allows for reduced connectivity among them. There still either needs to be a direct link or at least two paths between two parties. We cannot guarantee that this is true for all supply chains.

Reducing the necessary connectivity is necessary for supply chain privacy, but not sufficient. Secure computation protocols require the parties to agree on a joint function. This definition already leaks the secret supply chain visibility information. In our protocols only the adjacent parties need to agree on the functionality with the semi-trusted third party.

SMC protocols [2, 8], even when requiring less connectivity [5, 11, 16], require a binary circuit for the functionality. Let θ be the number of items to benchmark, then this circuit has a size of $O(\theta)$ for our metrics.

If we want to achieve supply chain privacy, a computation has to be performed for every possible item. We emphasize that this would be impractical even without security. Let θ' be the number of all possible items, i.e., the size of the domain of identifiers (96 bits in most existing systems). Let n be the number of stages in the supply chain and n' the total number of parties in the supply chain. Without optimizations such as computation servers, SMC protocols have a communication complexity of $O(n'^2\theta')$. Even with computation servers and other optimizations the communication complexity is at least $O(n'\theta')$ for supplying the input. Our protocols have a communication complexity of $O(n\theta)$.

5. CONCLUSION

We present a system for computing new metrics within RFID-supported supply chains without revealing the tracking data. We used techniques from Secure Multi-Party Computation (SMC), but designed special protocols that are adapted to the communication structure of a supply chain. Neither participation in the protocol nor the definition of the function violate industrial privacy. Our system is secure in the standard semi-honest model, but without collusion with the semi-trusted third party. We give standard proofs for our protocols and also prove the protocols are private in our definition of supply chain privacy. Furthermore, we significantly reduce the complexity compared to general SMC. We implemented the protocols and performance results are acceptable for practical application. All messages are limited in size and can be transported by storing them on the RFID tags used in the supply chain.

6. ACKNOWLEDGEMENTS

This work was partly funded by the European Commission under grant FP7-213531 to the *SecureSCM* project.

7. REFERENCES

- [1] 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*, 2008.
- [2] 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.
- [3] P. Bogetoft, D. Christensen, I. Damgard, M. Geisler, T. Jakobsen, M. Kroigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, T. Toft. Secure Multiparty Computation Goes Live. *Proceedings of the 13th International Conference on Financial Cryptography and Data Security*, 2009.
- [4] S. Chopra, and M. Sodhi. Looking for the Bang from the RFID Buck. *Supply Chain Management Review*. Available at <http://www.scmr.com/article/CA6444375.html>, 2007.
- [5] I. Damgard, and Y. Ishai. Constant-Round Multiparty Computation Using a Black-Box Psuedorandom Generator. *Proceedings of Crypto*, 2005.
- [6] J. Garay, and R. Ostrovsky. Almost-Everywhere Secure Computation. *Proceedings of Eurocrypt*, 2008.
- [7] O. Goldreich. Secure Multi-party Computation. Available at www.wisdom.weizmann.ac.il/~oded/pp.html, 2002.
- [8] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987.
- [9] W. Hedgepeth. RFID Metrics. *CRC Press*, 2007.
- [10] F. Kerschbaum. Practical Privacy-Preserving Benchmarking. *Proceedings of the 23rd IFIP International Information Security Conference*, 2008.
- [11] F. Kerschbaum. Adapting Privacy-Preserving Computation to the Service Provider Model. *Proceedings of the International Conference on Information Privacy, Security, Risk and Trust*, 2009.
- [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] Y. Lindell, B. Pinkas, and N. Smart. Implementing two-party computation efficiently with security against malicious adversaries. *Proceedings of the 6th Conference on Security and Cryptography for Networks*, 2008.
- [15] D. Naccache, and J. Stern. A New Public-Key Cryptosystem Based on Higher Residues. *Proceedings of the ACM Conference on Computer and Communications Security*, 1998.
- [16] M. Naor, B. Pinkas and R. Sumner. Privacy Preserving Auctions and Mechanism Design. *Proceedings of the 1st ACM Conference on Electronic Commerce*, 1999.
- [17] M. Naor, and B. Pinkas. Efficient Oblivious Transfer Protocols. *Proceedings of the Symposium on Data Structures and Algorithms*, 2001.
- [18] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Proceedings of EUROCRYPT*, 1999.
- [19] B. Santos, and L. Smith. RFID in the Supply Chain: Panacea or Pandora's Box? *Communications of the ACM 51(10)*, 2008.
- [20] S. Sarma, D. Brock, and D. Engels. Radio frequency identification and the electronic product code. *IEEE Micro 21(6)*, 2001.
- [21] A. Yao. Protocols for Secure Computations. *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1982.