Secure Product Tracking In Supply Chain

Mehdi Khalfaoui[1], Refik Molva[2], and Laurent Gomez[1]

[1] SAP Labs France, Mougins 06254 France,
mehdi.khalfaoui@sap.com, laurent.gomez@sap.com
[2] Eurecom Institut, Biot 06410 France,
refik.molva@eurecom.fr

Abstract. In this paper, we propose a secure and efficient product tracking mechanism implemented using wireless sensor nodes. This mechanism aims at tracking goods, and actions performed by each actor of the supply chain, while preserving the actors' privacy. Our solution is based on generating identifiers of the actors and their activities using AND Anti collusion codes. The identifiers are encrypted using homomorphic encryption to ensure security against adversaries, and to be able to compress the collected identifiers. In this work, wireless sensor nodes are not required to perform complex computation, which makes our solution feasible.

Keywords: Supply chain; Sensor networks; Tracking; Privacy; AND-ACC; Homomorphic Encryption

1 Introduction

This paper aims at introducing a secure and privacy preserving product tracking solution using wireless sensor networks into supply chains. To this effect, we use sensors as secure storage tokens to store the steps of the path. More precisely, the memory capacity of the sensor nodes that are attached to the product store the identities of the involved steps. At each step, the sensor updates its memory to add the current step identity. The set of steps collected by a sensor is the path trace that the product went through. Each step has a secure identity, in order to be identified and to prevent impersonation attacks. At the end of the supply chain, a verifier extracts the steps and verifies their validity. The verifier is often a supply chain manager. He wants to have a global overview of its supply chain in order to mitigate any potential threat.

In order to preserve the actors privacy, the steps' identities should be kept secret. Encryption appears to be a straightforward solution preventing eavesdroppers from stealing identities and impersonating any legitimate supply chain actor. However, any technical solution that addresses secure and privacy preserving product tracking should take into account the limitations of sensor nodes. These are constrained devices in terms of computation power, memory, and energy. Due to the limited memory featured by sensors, straightforward storage of the collected steps cannot be afforded. Thus, a compression mechanism to

reduce the size of the path traces is a mandatory requirement in order to overcome the memory limitation of sensors. Also, the limited power computing and the energy of the sensor nodes, make the implementation of complex functions such as public key encryption algorithms difficult. Therefore, any operations that have to be performed by the sensors should be compatible with such constraint environment.

This paper introduces a mechanism to track products in supply chain while protecting sensitive information of supply chain actors and products. The main idea is to store on the sensor node an encrypted and compressed path trace for tracing the product. For that purpose, we propose to generate identities for the different steps using AND-Anti Collusion Code (AND-ACC) [19]. This code can be easily compressed by construction. Furthermore, to assure privacy, steps' IDs should be protected using an encryption mechanism that is compatible with the compression technique. Therefore, homomorphic encryption seems to be a natural choice. In addition, to ensure the legitimacy of the involved actors, sensors use Rabin scheme to authenticate them. The main features of the suggested product tracking scheme are as follows:

- It allows the supply chain manager to verify the legitimacy of the path taken by a product. More precisely, it allows the supply chain manager to verify which set of steps, a product has visited.
- It guarantees the privacy of products and therewith partners in the supply chain. Only the supply chain manager is able to verify the path taken by a product.
- It allows the restriction of information available to each supply chain actor, such as the origin of the product and the final destination.

Moreover the scheme is suitable for low capacity sensors. It only requires a few Kbytes storage. The protocol execution for each supply chain step requires only two modular multiplication.

2 Related work

The idea of our use case that WSN can be used for tracking the goods in the supply chain was first suggested in [1]. However, research focuses mainly on Tag RFIDs to achieve a secure supply chain. Ouafi and Vaudenay [11] address counterfeiting of products using strong cryptography on RFID tags. Elkhiyaoui et al. [1] presented a tracker, a new mechanism to protect against malicious state update of tags in each step of the supply chain. Secure tracking of specific target using WSAN was also addressed in [5]. It describes a mechanism of tracking a moving target based on relaxation algorithms [13]. However, passive RFID tags have limited resources, which make security hard to achieve. As a matter of fact, any public key cryptosystem cannot be used with this type of RFID tag. Only hash functions can be implemented and used in passive RFID tag environment. Chawla et al. [2] check whether covert channels exist in a supply chain that leak

information about a supply chains internal details to an adversary using security mechanism implemented in RFID tag. Therefore, a tag's state is frequently synchronized with a backend database. If a tag's state contains data that is not in the database, the tag is rejected. Our mechanism's focus, however, is on the secure, privacy-preserving detection of which path a tag has taken. Shuihua and Chu [15] detect malicious tampering of a tags state in a supply chain using watermarks. However, there is neither a way to identify a tag's path, nor to protect its privacy in the supply chain. Kerschbaum and Oertel [9] detect counterfeits in the supply chain using pattern matching for anomaly detection. When a tag is read, this information is stored in a central database along with the ID of the tag. Unlike our mechanism, the focus of this paper is on the privacy-preservation of readers participating in the supply chain. There is no privacy for the tags in the supply chain. Regarding simple product genuineness verification, solutions exist that rely on physical properties of a "tag". For example, TAGSYS produces holographic "tags" that are expensive to clone [18]. Verayo produces tags with Physically no clonable Functions (PUF) [20]. While these approaches solve product genuineness verification, they neither support identification of tags paths nor any kind of privacy properties. Our construction based on anti collusion code can be similar to collusion detection in multimedia files using fingerprints [19]. It allows a privacy preserving and the anonymity of the supply chain actors. Furthermore, a sensor node has more resources , which enables the use of more advanced cryptographic tools.

3 Background

Supply Chain is the movement of materials as they flow from their source to the end customer. Supply Chain includes purchasing, manufacturing, warehousing, transportation, customer service, demand planning , supply planning and supply chain management. It is made up of people, activities, information and resources involved in moving a product from its supplier to customer.
Formally, a supply chain is represented by a digraph $G = (V, E)$ whereby each vertex v represents one step in the supply chain. A step v in the supply chain is uniquely associated with an entity.
Each directed edge e, which links vertex v_i to vertex v_j , express that v_j is a possible next step to step v_i in the supply chain. This simply means that according to the organization of the supply chain, a product might proceed to step v_j after the completion of step v_i. Note that a supply chain can include loops and reflexive edges, but for the sake of simplicity, we assume that in our system there are no loops or reflexive edges. Whenever a product in the supply chain proceeds to step v_i , the entity interacts first with the sensor. A path P is defined as finite sequence of steps $P = v_0, ..., v_l$, where l is the length of the path P. A Path P is deemed valid if it is part of a legitimate supply chain networks.

3.1 Entities

A Secure Product Tracking (SPT) system consists of the following components:

- **Sensor** S: each product in the supply chain is equipped with a sensor. A sensor S is a re-writeable memory that stores the sensor state $s_{(S,v_j)}$, where v_j represents the current step that is being visited by S.
- **Supply chain manager** M: M is in charge of the the initial setup of sensors and of the verification of the path taken by each sensor. In order to verify the path of sensor S, M reads the current state $s_{(S,v_j)}$ of S, and decides whether the sequence visited by S is legitimate in the supply chain. We assume that M can enumerate all the valid paths in the supply chain.
- **actor** a: each actor is a legitimate single entity of the supply chain. When a product visits a step, the actor associated with that step interacts with the sensor S attached to the product.
- **trace** T: T is a digest of the path taken by a product and stored in the sensor attached to it.

Thus, a SPT system is:

- a Supply Chain $G = (V, A)$
- a Sensor S
- a set of possible traces $\mathcal{T}$
- a set of different actors $\dashv$
- a supply chain manager M
- a set of valid paths $\mathcal{P}$

4 Adversary Model

In our protocol, we assume that actors are semi-honest. That is, actors generate well formed protocol messages but they potentially can forge paths, impersonate other actors, or try to retrieve the identity of the steps encoded in a path trace. Any adversary can read the sensor's memory, since sensors are not tamper-resistant. However, a sensor can only update its trace after a successful authentication. Therefore, only the legitimate supply actors can make sensor updates to its path trace. The security and the privacy of the protocol will be evaluated based on a formal adversary model inspired by Valbuany et al. [11]. The adversary $\mathcal{A}$ that aims at violating the security and privacy properties, relies on the following oracles:

- $\mathcal{O}_{Choose}$: picks a sensor from the supply chain.
- $\mathcal{O}_{Read}$: takes a sensor S as input, and reads its trace.
- $\mathcal{O}_{Send}$: takes a sensor S as input, and sends a trace to it.
- $\mathcal{O}_{Check}$: returns $TRUE$ if the sensor S went through a valid path, otherwise it returns $FALSE$.
- $\mathcal{O}_{Inject}$: injects a sensor S in the supply chain.

We describe the security and privacy properties of our protocol as follows:

4.1 Security

Our protocol aims at preventing path forgery by an adversary by assuming the following property: if the verification of a sensor's trace T by the supply chain manager M returns a valid path P_v, then S must have visited all the steps of the path P_v.
In the rest of this section, we introduce two phases, Learning phase when $\mathcal{A}$ can query the oracle $\mathcal{O}_{Choose}$, that randomly selects a sensor within the supply chain and gives it to $\mathcal{A}$. During this phase, $\mathcal{A}$ is allowed to read traces stated in S and sends traces to S using the oracles $\mathcal{O}_{Read}$ and $\mathcal{O}_{Send}$ respectively. Then, $\mathcal{A}$ checks the validity of the trace in the sensor S, by querying the oracle $\mathcal{O}_{Check}$.

In the challenge phase, $\mathcal{A}$ randomly chooses a sensor S that has been already deployed in the supply chain, and authenticates itself to it. Then, $\mathcal{A}$ sends some value to S using $\mathcal{O}_{Send}$. Therefore, $\mathcal{A}$ has to successfully bypass the authentication process, to make S update its path trace. Finally, $\mathcal{A}$ injects the sensor in the supply chain using $\mathcal{O}_{Inject}$. If M accepts the sensor as a valid one, $\mathcal{A}$ wins.

For Cloning attacks, We use the same mitigation technique presented by Elkhiyaoui et al. [1]. M has a database DB_c of the sensors that went through valid path of supply chain, and were verified correctly by M. Therefore, $\mathcal{A}$ cannot clone a sensor more than once, thus, the cloning cannot be performed in large scale.

4.2 Privacy

An adversary $\mathcal{A}$ in our protocol, beside his ability to eavesdrop the sensor's communications with the actors' systems, she is able to tamper with the sensor's memory as well. Thus, we identify privacy requirement as step unlinkability. Step unlinkability prevents $\mathcal{A}$ from telling that two different sensors interacted with a common step.

To have a formal definition, we introduce the following oracles into our adversary model:

- $\mathcal{O}_{distinguish}$: takes as input two states $s_{(S_i,v_k)}$ and $s_{(S_j,v_k)}$, and returns $TRUE$ if the S_i, and S_j refer to the same sensor.
- $\mathcal{O}_{step}$: takes as input a sensor S and step v, and returns $TRUE$ if the S went through step v, and false otherwise.

the privacy game has two phases, learning phase and challenge phase. In the learning phase, $\mathcal{O}_{choose}$ provides to $\mathcal{A}$ a list of sensors. $\mathcal{A}$ can then observe the numerous protocol exchanges and collect sequences of valid paths. In the challenge phase, after several additional interactions, a random sensor Sc, that is already provided to $\mathcal{A}$ in the learning phase, is chosen. If $\mathcal{A}$ is able to distinguish with high probability which sensor from its list corresponds to the sensor challenger, $\mathcal{A}$ wins.

5 Protocol Description

5.1 Approach

In this section, we introduce our approach, and the different requirements that our solution has to fulfill.
The main idea of this work is to use the sensors as secure storage tokens for the path traces. More precisely, sensor node S is attached to the goods along the supply chain. At each step, S collects the current step's identity. S keeps track of which step interacted with the goods and which activity has been performed. The activity can be any process that the actor is supposed to perform, such as transportation or delivery.

There are three phases in our solution: *Initialization*, *Collection*, and *Verification*.

- *Initialization*: M generates the steps' IDs. Then, M distributes each step's ID to the corresponding supply chain actor.
- *Collection*: IDs are kept secret by the actors. During the course of regular supply chain operations, when a product is handled by a supply chain actor, the sensor S that is attached to the product and the actor interact through this phase of the protocol. As part of this phase, S verifies the legitimacy of the supply chain actor. If the actor is legitimate, S includes the current step's ID in the path trace.
- *Verification*: At the end of the supply chain, M retrieves the path trace from S and extracts the steps' IDs from the trace and verifies their legitimacy.

In addition to the security and privacy properties introduced in section 4, this approach requires the actors' IDs to be kept secret. Furthermore, the solution has to take into account the limitations of sensors in terms of computational power and memory. Therefore, straightforward storage of actors IDs and complex cryptographic operations such as asymmetric encryption cannot be afforded by sensors.

The first objective of the solution thus is to come up with a data compression technique that allows to store a digest of the path in the sensors. This technique should also allow the retrieval of actors IDs form the trace by M. The second requirement is for the confidentiality of the content of the path namely, the actor ID's included therein. Encryption that appears to be the most suitable solution to meet that requirement has to comply with the compression technique. Moreover, to allow the integration of encrypted actor ID's into the path digest without decrypting the former, homomorphic encryption seems to be an appropriate solution to this question.

To build a scheme that satisfies the aforementioned requirements, we leverage on a number of well established primitives. As for the compression technique, our solution relies on Anti-Collusion Code (ACC). We leverage on Paillier encryption

to enable S to aggregate the encrypted IDs without decrypting them. Finally, we use polynomial conversion in order to convert a ACC code vector to a polynomial evaluation of a specific value. This conversion allow to encrypt a value instead of a vector without information loss.

5.2 Preliminaries

Anti Collusion Code

Definition A binary code $C = \{c_1, c_2, \ldots, c_n\}$ *such that the logical AND of any subset of k or fewer code vectors is non-zero and distinct from the logical AND of any other subset of k or fewer code vectors is a k-resilient AND anti-collusion code, or an AND-ACC code.* [19]

Such code is often used in digital fingerprinting to prevent collusion attacks against traditional watermarking techniques [10]. It allows the identification of groups of K or less colluders. This is similar to our case, since the supply chain actors could be considered as colluders that collaborate to perform specific actions on the product. Therefore, each supply chain actor can be associated to a code vector. The result of the bitwise AND operation between the code vectors marks the path of the product. The bitwise AND operation does not increase the size of the path, therefore, the path size can stay manageable by the sensor all along the supply chain. Also, encoding and decoding operations does not require a lot of resources.

Encoding To encode up k code vectors in a single code vector c_{enc}, we perform bitwise AND operations on the input code vectors.

$$c_{enc} = c_1 \text{ AND } c_2 \ldots \text{ AND } c_l \tag{1}$$

Decoding To decode c_{enc}, we extract the positions of c_{enc} coefficients that equal to 1. All the code vectors c_i that have their coefficients at the extracted positions equal to 1 are the ones encoded in the c_{enc}.

Example The columns of the following matrix form a 2-resilient AND-ACC code

$$\begin{pmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \\ 0&1&1&1&1&0&0 \\ 1&1&0&0&1&1&0 \\ 1&1&0&1&0&0&1 \end{pmatrix}$$

This code requires 7 bits for 7 users and provides 2-resiliency since any two column vectors share a unique pair of 1 bits. The encoding result of c_1 and c_7 is $c_{enc} = c_1$ AND $c_7 = \{0,0,1,0,0,1\}$.

Based on the decoding technique c_{enc} has 1's at positions 3 and 7. Thus, the decoding of c_{enc} yields c_1 and c_7 since these are the only two code vectors with 1's at these positions.

Paillier Cryptosystem Paillier cryptosystem [12] that we use in order to encrypt the steps' IDs, has an interesting property:

Additive Homomorphic property This property allows us to compute the encrypted sum of two or several encrypted values as follows:

$$\mathcal{E}(m_1, r_1) * \mathcal{E}(m_2, r_2) = \mathcal{E}(m_1 + m_2, r_1 r_2) \tag{2}$$

Rabin Scheme Rabin cryptosystem [14] is used to achieve authentication of the supply chain actors. Rabin encryption is single square modular encryption, which makes it feasible for low capacity devices such as sensor nodes. Rabin's public key $n_R = p_R q_R$ is stored in the sensor to perform the authentication with the visited supply chain actors. The public key is necessary for later encoding and can be published, while the private key must be possessed only by the recipient of the message.
The key-generation process can be summarized as follows:

- Choose two large distinct primes p and q.
- Let $n_R = p \cdot q$. Then n is the public key. The pair (p, q) is the private key.

Encryption using the public key n_R, the plaintext is encrypted as follows:

$$\mathcal{R}(m) = m^2 \, mod \, n_R \tag{3}$$

Decryption Using the private key (p,q), the decryption operation requires the solution of

$$x^2 = \mathcal{R}(m) \, mod \, N \tag{4}$$

which is "easy" if the factors of n_R are known. This equation has four roots in $\mathcal{Z}^*_{\mathcal{N}_\mathcal{R}}$. The solution is the one that fulfills the following additional requirement:

$$x \quad mod \quad 2 = 0 \quad and \quad ((xmodp) + (xmodq)) \quad mod \quad 2 = 0 \text{ [14]} \tag{5}$$

Put it all together AND-ACC allows to encode the path, which is done by assigning to each supply chain actor a code vector as identifier. The collected identifiers are encoded into a single value using bitwise AND operation. However, as the actors' identities have to be kept secret, thus to be encrypted before getting collected. Therefore, an encryption mechanism that is homomorphic with bitwise AND is required to ensure the confidentiality of the actors' IDs.

Definition $\mathcal{E}$ is homomorphic with respect to bitwise AND operation, if and only if there is an operation *op*, which verifies the following: for any pair of messages m_1 and m_2, we have:

$$\mathcal{E}(m_1 AND m_2) = \mathcal{E}(m_1) \, op \, \mathcal{E}(m_2) \tag{6}$$

Unfortunately, there is no such encryption mechanism that is homomorphic with respect to bitwise AND operation. The only homomorphic encryption that we know, are homomorphic with respect to the multiplication such as the RSA cryptosystem

$$m_1{}^e \cdot m_2{}^e = (m_1 \cdot m_2)^e$$

or homomorphic with respect to the addition such as Paillier cryptosystem

$$g^{m_1} \cdot g^{m_2} = g^{(m_1+m_2)}$$

So ideally, we need a way to associate bitwise AND operation with arithmetic addition or multiplication, in order to be able to use one of the existing homomorphic encryption algorithms.

AND-SUM conversion We decided to apt for a conversion between bitwise AND operation and arithmetic addition is based on the following property:

Property 1 Let's $x_1, \ldots, x_n$ be n binary values, we have the following equivalences of Table 1:

$$x_1 + \ldots + x_1 = n \iff x_1 = \ldots = x_n = 1 \iff x_1 \wedge \ldots \wedge x_n = 1 \tag{7}$$

Table 1. AND-SUM conversion

addition $+$	AND $\wedge$
$\sum_{i=0}^{n} x_i = n$	$\wedge_{i=0}^{n} x_i = 1$
$\sum_{i=0}^{n} x_i < n$	$\wedge_{i=0}^{n} x_i = 0$

By interpreting binary code vectors as arithmetic ones, we can sum them as follows:

$$c_{sum}[j] = \sum_{i=0}^{n} c_i[j] \tag{8}$$

Thanks to Property 1, the expected result of the AND bitwise operation $c_{enc} = \wedge_{i=0}^{n} c_i$, can be retrieved on simple reasoning. Indeed, the $\max_{i=0}^{length(c_{sum})} c_{sum}[i]$ corresponds to the number of the input code vectors. Thus,

$$if c_{sum}[j] = \max_{j=0}^{length(c_{sum})} c_{sum}[j] \; then \; c_{and}[j] = 1 \tag{9}$$

$$if c_{sum}[j] < \max_{j=0}^{length(c_{sum})} c_{sum}[j] \; then \; c_{and}[j] = 0 \tag{10}$$

AND-ACC operations on encrypted code vectors Now that the basic encoding of actors' IDs can be represented based on the sum of code vectors in AND-ACC code, the additively homomorphic Paillier encryption seems to be a suitable solution to assure the privacy of steps' IDs as part of the scheme. Thus, each code vector c_i is encrypted using Paillier cryptosystem $\mathcal{E}$

$$\mathcal{E}(c_i) = \begin{pmatrix} \mathcal{E}(c_i[0]) \\ \mathcal{E}(c_i[1]) \\ \dots \\ \mathcal{E}(c_i[len]) \end{pmatrix} \tag{11}$$

The encrypted path can then be computed using the additively homomorphic property of Paillier, as follows:

$$\mathcal{E}(Path_{sum}) = \prod_{i=0}^{len} \begin{pmatrix} \mathcal{E}(c_i[0]) \\ \mathcal{E}(c_i[1]) \\ \dots \\ \mathcal{E}(c_i[len]) \end{pmatrix} \tag{12}$$

$$\mathcal{E}(Path_{sum}) = \begin{pmatrix} \mathcal{E}(\sum_{i=0}^{len} c_i[0]) \\ \mathcal{E}(\sum_{i=0}^{len} c_i[1]) \\ \dots \\ \mathcal{E}(\sum_{i=0}^{len} c_i[len]) \end{pmatrix} \tag{13}$$

An encrypted path is decrypted as follows:

$$Path_{sum} = \mathcal{D}(\mathcal{E}(Path_{sum})) = \begin{pmatrix} \sum_{i=0}^{len} c_i[0] \\ \sum_{i=0}^{len} c_i[1] \\ \dots \sum_{i=0}^{len} c_i[len] \end{pmatrix} \tag{14}$$

Finally, the actual cleartext value of the path encoding is derived from $Path_{sum}$ using the reverse conversion as described in 5.2.

Due to separate encryption of each coefficient in the code vectors, the path encoding scheme still would require large memory space to store a single path trace. The path encoding scheme, thus, involves an additional technique that allows us to represent code vectors as simple integers.

Vector-integer conversion In order to convert a vector $v = (v_0, \dots, v_l)$ to an integer, we consider the coordinates of v as coefficient of polynomial P. We choose a value x bigger than $k + 1$ (k is the parameter such as AND-ACC is

K-resilient), and we compute Poly$(v)(x)$= $\sum_{i=0}^{len} v_i x^i$. where len is the length of v.

The inverse conversion is easily achieved by means of subsequent divisions to extract the coefficients of the polynomial, which are the coefficients of the vector by construction.

It should be noted that the sum of polynomial evaluation representing two code vectors is identical to the polynomial representation of the sum of the two code vectors.

5.3 SPT Protocol

Based on the aforementioned building blocks, the operations of the SPT are depicted in three phases.

Initialization SPT's setup is as follows:

- Supply chain manager M shares with the supply chain actors a Rabin's private key (p_R, q_R). Then, M stores the public key n_R in the sensor.
- M generates a Paillier cryptosystem public and private keys. M publishes the Paillier's public key to all the supply chain actors.
- M generates randomly a list of ACC codes v using Generate algorithm. Then, M converts each code vector v to a value Poly$(v)(x)$, which corresponds to an identity of the supply chain step.
- M encrypts each value $Poly(v_i)$ to get $\mathcal{E}(a_i)$, using Paillier's public key. Then, M sends the encrypted value to the corresponding a_i through a secure channel.
- M generates an s_{id} for each sensor S, and key k for keyed-HMAC computation. Then, M computes the *encrypted hash* $\mathcal{E}(HMAC_k(s_{id}))$. M keeps a database DB_sensor of sensor identities s_{id}, and their hashes $HMAC_k(s_{id}))$.
- M stores in the sensor S, Rabin's public key n_R, the keyed-HMAC $HMAC_k(s_{id}))$, and the initial value $\mathcal{E}(HMAC_k(s_{id}))$. $\mathcal{E}(HMAC_k(s_{id}))$ corresponds to the initial trace stored on the sensor.
  Now, for each sensor entering the supply chain, M has already stored on it the initial value $s_{(T_0)} = E(HMAC_k(s_{id}))$.

Collection This phase starts when S arrives to the supply chain actor. Here, we assume that the sensor S has visited the steps $step_0, \ldots, step_l$. When, S visits the step $step_{l+1}$, it has already stored the trace of the path $\mathcal{P}_l = \overrightarrow{step_0 step_1 ... step_l}$. Therefore, the current path trace that is stored in the sensor is $\mathcal{E}(T_l)$, which corresponds to the state of the sensor after visiting l steps.

There are two sub-phases, authentication sub-phase to check the legitimacy of the actor, and the trace collection sub-phase to update the path trace.

In the authentication sub-phase, S chooses a random value $r \in \mathcal{F}_{n_R}$ and sends $Rabin(r) = r^2 mod\, n_R$ to the current actor. This latter decrypts $Rabin(r)$ using its private key, and returns $hash(r)$. S considers the authentication as

successful, if the received value matched the hash value of the generated one. Then, S can proceed to trace collection phase.

In the collection sub-phase, S sends its $HMAC_k(s_{id}) \oplus r$ to the current actor. This latter computes $\mathcal{E}(Poly(v_{l+1}(x)))^{HMAC_k(s_{id})} = \mathcal{E}(Poly(v_{l+1}(x)) * HMAC_k(s_id))$, and he sends to S, the value $\mathcal{E}(Poly(v_{l+1}(x)) * HMAC_k(s_{id})) \oplus r$. S retrieves $\mathcal{E}(Poly(v_{l+1}(x)) * HMAC_k(s_id))$, and updates the path trace $\mathcal{E}(T_{l+1})$ as follows:

$$\mathcal{E}(T_{l+1}) = \mathcal{E}(T_l) * \mathcal{E}(Poly(v_{l+1}(x)) * HMAC_k(s_{id}))\ mod\ N^2 \tag{15}$$

$$\mathcal{E}(T_{l+1}) = \mathcal{E}(T_l + Poly(v_{l+1}(x)) * HMAC_k(s_{id}))\ mod\ N^2 \tag{16}$$

$$\mathcal{E}(T_{l+1}) = \mathcal{E}((\sum_{j=0}^{len-1} (\sum_{i=0}^{l+1} v_i[j]) x^j) * HMAC_k(s_{id}))\ mod\ N^2 \tag{17}$$

Table 2 illustrates the exchanged messages between S and the current actor $a_{current}$.

Table 2. Authentication phase

$\mathcal{S}$ picks randomly a number r
$\mathcal{S} \longrightarrow a_{current} : r^2 mod N_R$
$a_{current} \longrightarrow S : hash(r)$
$a_{current} \longrightarrow S : \mathcal{E}(Poly(v_{l+1}(x))) \oplus r$

Verification Using the authentication protocol presented in the collection phase, M authenticates the sensor S. Then, M verifies if $HMAC_k(s_{id})$ is in DB_{sensor} to check the legitimacy of the sensor and to prevent massive cloning attacks. If $HMAC_k(s_{id})$ exists in DB_{sensor}, M accepts the path trace T_m from S, otherwise it rejects it. M decrypts the path trace using the secret key of Paillier cryptosystem. the decryption result is $T_m = HMAC_k(s_id) * \sum_{1 \leq i \leq n} Poly(v_i)(x)$.

M checks if T_m is well formed by verifying $T_m\ mod\ HMAC_k(s_{id}) = 0$. If it is the case, M extracts the coefficients of the polynomial, using simple euclidean division operations to have the vector c_{vec} which is the sum of the coordinates of the different actors' identifiers.

Table 3 shows the algorithm to extract the value of the vector c_{vec}.

Using the mechanism explained in section 5.2, we convert c_{vec} to its corresponding binary vector c_{enc}.

c_{enc} represents AND bitwise operation result for the different vectors v_i. M decodes c_{enc} , and extracts the involved steps. Then, M checks if the path is a valid one or not.

Table 3. c_{vec} extraction algorithm

Let $qot(a, b)$ the quotient of the division of a by b
$qot(T_m, HMAC_k(s_id)) = \sum_{0 \leq i \leq l} Poly(v_i)(x)$
$qot(T_m, HMAC_k(s_id) = \sum 0 \leq i \leq l \sum 0 \leq j \leq len-1 \; v_i[j]x^j$
$c_{vec}[i] = \sum 0 \leq j \leq len-1 \; v_i[j]$

In this section, we presented our solution in detail. We demonstrated that only a two modular multiplication operations are executed in the sensor node per step. Furthermore, our solution can be used for path enforcement by a supply chain manager, or by a high authority to trace back the actors that interacted with the product. In Section 6, we present a provable security to our approach. We describe our simulator to show that any adversary who is able to break our system, she can solve the hard problem of the quadratic residue problem [12].

6 Security Analysis

In this section we prove the security of SPT system. The proof was inspired from [1] [11].

6.1 $\mathcal{A}$ is not a legitimate actor

if $\mathcal{A}$ authenticates successfully itself to S, she breaks the Rabin scheme security by definition. Therefore, only the legitimate supply actors can try to update the path trace maliciously.

6.2 $\mathcal{A}$ is a legitimate actor

If $\mathcal{A}$ is legitimate actor, the authentication process will succeed, and S accepts the received value from $\mathcal{A}$. In this case, we use the security of keyed-HMAC and the decisional composite residuosity assumption to prove the security of SPT protocol against forgery by a legitimate actor.

Security of keyed-HMAC For our proof sketch, we are using the property indistinguishability of keyed hash function.

Indistinguishability property Let $\mathcal{O}_{distinguish}$ be an oracle that when $\mathcal{A}$ provides it with a message m, $\mathcal{O}^{distinguish}$ returns with the same probability a random number, or $HMAC_k(m)$. $\mathcal{A}$ cannot guess with no-negligible probability if the returned value is a random number, or $HMAC_k(m)$.

Lemma Producing a new valid trace contradicts the indistinguishiability property of $HMAC_k$.
Proof (Sketch). From $\mathcal{A}$, we can build an adversary $\mathcal{A}'$ that uses $\mathcal{A}$ to break the indistinguishability property of keyed-HMAC. we provide $\mathcal{A}$, with a sensor S and its s_{id}. $\mathcal{A}$ produces a new valid trace $\mathcal{E}(T_m)$ that corresponds to the sensor s_{id}. $\mathcal{A}$ provides s_{id} to $\mathcal{O}_{distinguish}$. $\mathcal{O}_{distinguish}$ returns value H to be tested. $\mathcal{A}$ decrypts $\mathcal{E}(T_m)$. She gets T_m, and computes $T_m \, mod \, H$. If $T_m \, mod \, H = 0$, H is the $HMAC_k(s_id)$, otherwise H is a random number.

The decisional composite residuosity assumption (DCRA) The DCRA states that given a composite n and an integer z, it is hard to decide whether z is a n-residue mod n^2 or not. In other words, whether there exists y such that $z = y^n mod \, n^2$. This assumption is mainly used to proof the semantic security of Paillier cryptosystem [12].

Definition 1. *A cryptographic protocol is semantically secure if its indistinguishability against chosen plaintext attacks (IND-CPA) holds.*

Theorem 1. *SPT is semantically secure if and only if DCRA and the indistinguishibilty of keyed-HMAC hold.*

Proof. The main idea of this proof is to build an attacker $\mathcal{A}\prime$ from $\mathcal{A}$ whose advantage ϵ to forge a valid path, that is able to break DCRA. As shown in the previous *Lemma*, $\mathcal{A}$ cannot provide a new valid path trace from scratch. Now, Let's assume that $\mathcal{A}$ can update a valid path trace that she got from the learning phase to a new valid path trace. For the sake of simplicity, we consider that the SPT system has only one valid path.

Let $\mathcal{O}_{DCRA}$ be an oracle that, when it is queried with a parameter n, it flips a coin $b \in 0, 1$. If $b = 1$ it takes a $y \in Z$ and returns $y^n mod \, n^2$ otherwise, it returns a random number C.

$\mathcal{A}\prime$ creates SPT system with a valid path, ($step_0$, ..., $step_m$). Then, she generates the AND-ACC identifier corresponds to each step.
Let $\mathcal{E}(T_{m-1})$ be the encrypted path trace until the $step_{m-1}$. First, $\mathcal{A}'$ sends a query to $\mathcal{O}_{DCRA}$ with N (Paillier modular) as parameter, and gets a challenge C. $\mathcal{A}'$ computes $\mathcal{E}(T_{m-1})$*C mod N^2, and writes the result in a sensor S.
If in the challenge phase, $\mathcal{A}$ is able to update the trace to a valid path trace $\mathcal{E}(T_m)$, then $\mathcal{E}(T_{m-1})$. C mod N^2 is a valid ciphertext of T_{m-1} (i.e $\mathcal{E}(T_{m-1})$. C mod N^2 is re-encryption to $\mathcal{E}(T_{m-1})$). Therefore, C is N-residue mod N^2, and $\mathcal{A}'$ breaks the DCRA assumption, with advantage of 1/2*ϵ, since she is wrong half of the time because of oracle's coin flip.
Table 4 illustrates the messages exchanged between the involved entities during the challenge game.

7 Privacy analysis

In this section we prove the privacy requirement of step unlinkability of SPT system.

Table 4. Forgery challenge game

$\mathcal{O}_{DCRA}$		$\mathcal{A}'$		$\mathcal{A}$
receive N	$\longleftarrow$	send N		
pick C	$\longrightarrow$	receive C		
		compute $\mathcal{E}(T_{m-1}).C mod\, N^2$	$\longrightarrow$	receive $\mathcal{E}(T_{m-1}).C mod\, N^2$
		receive $\mathcal{E}(T_m)$	$\longleftarrow$	update trace $\mathcal{E}(T_m)$
receive 1 or 0 $\longleftarrow$ if $\mathcal{E}(T_m)$ is valid, send 1 else send 0				

Theorem 2. *SPT provides step unlinkability under DCRA.*

Proof. Assume there is an adversary A whose advantage ϵ to break the step unlinkability experiment is non-negligible. We now construct a new adversary $\mathcal{A}'$ that executes $\mathcal{A}$ and breaks the semantic security of Paillier.

Let $\mathcal{O}_{DCRA}$ be an oracle that, when it is queried with a parameter n, it flips a coin $b \in 0, 1$. If $b = 1$ it takes a $y \in Z$ and returns $y^n mod\, n^2$ otherwise, it returns a random number C.

$\mathcal{A}\prime$ creates SPT system with multiple valid paths. Then, She generates the AND-ACC identifiers correspond to each step. First, $\mathcal{A}'$ sends a query to $\mathcal{O}_{DCRA}$ with N (Paillier modular) as parameter, and gets a challenge C. Then, $\mathcal{A}'$ builds two traces for two different path, with one step in common. Let $\mathcal{E}(T_m)$ the path trace for the path ($step_0$, ..., $step_m$), and $\mathcal{E}(T'_m)$ the path trace for the path ($step'_0$, ..., $step'_m$) with $step_i$ and $step'_i$ are the common step. However, the identifier is $\mathcal{E}(v_i)$ for $step_i$ and $\mathcal{E}(v_i)$*C mod N^2 for $step'_i$. $\mathcal{A}'$ provides the two traces to $\mathcal{A}'$ in the challenge phase.

If in the challenge phase, $\mathcal{A}$ is able to decide if both traces has a common step with an advantage ϵ, then $\mathcal{E}(v_i)$. C mod N^2 is a valid ciphertext of v_i). Therefore, C is N-residue mod N^2, and $\mathcal{A}'$ breaks DCRA, with advantage of 1/2*ϵ, since She is wrong the half of the time because of oracle's coin flip.
Table 5 illustrates the messages exchanged during the challenge game.

8 Performance analysis

This section is allotted to present the analytical performance evaluation of the proposed scheme. First, we evaluate the AND-ACC code reduction performance, then the performances related to the sensor itself. The performance evaluation

Table 5. step unlinkability challenge game

$\mathcal{O}_{DCRA}$		$\mathcal{A}'$		$\mathcal{A}$
receive N	$\longleftarrow$	send N		
pick C	$\longrightarrow$	receive C		
		compute $\mathcal{E}(T_m)$ and $\mathcal{E}(T'_m)$	$\longrightarrow$	receive $\mathcal{E}(T_m)$ and $\mathcal{E}(T'_m)$
		receive 1 or 0	$\longleftarrow$	check if $\mathcal{E}(T_m)$ and $\mathcal{E}(T'_m)$ have a common step
receive 1 or 0 $\longleftarrow$ if 1 is received, send 1 otherwise send 0				

criteria of the sensor, are the storage cost, the computation cost and the communication cost.

In this paper, a Rabin's public key has a size of 1024 bits. The hash function used is SHA1, which has an output's size of 160 bits. Paillier encryption has an output's size of 2048 bits. For sensor identity, a size of 160 bits is chosen. Rabin's encryption is a single modular square which is considered equivalent to a single modular multiplication in this paper. It requires roughly $100\mu J$ using ATmel128 microprocessor [7] based on the result of Gaubatz et al. [4]. In our scheme, we use TinyRNG [3] to generate random numbers. TinyRNG consumes around $58\mu J$ at each random number generation. Hash function consumes roughly $1\mu J$ [6]. The communication cost are set to e_s =$0.209\mu J$ and e_r =$0.226\mu J$ from the characteristics of the CC2420 transceiver used in the Xbows MICA-Z and Telos B sensor nodes [8].

8.1 AND-ACC reduction cost

For the sake of simplicity, we consider that our supply chain has only one valid path. Let l be the number of steps in our supply chain. Let (C) consist of all l-bit binary vectors that have only a single 0 bit. For example, when $l = 4$, $(C) = \{1110; 1101; 1011; 0111\}$. It is easy to see when $k \leq l - 1$ of these vectors are combined under AND, that this combination is unique. This code has cardinality d, and can produce at most d vectors.

The encrypted path trace $\mathcal{E}(T)$ is valid only if the size of T is less than 1024 bits. Therefore, the trace $T = \sum_{i=0}^{l} v_i x^i$ of the path $\{step_0, ..., step_{l-1}\}$ has to have a size less than 1024 bits. As mentioned in section 5.2, $x \geq l$, thus:

$$T \leq \sum_{i=0}^{l} v_i l^i \leq \sum_{i=0}^{l})l * (l+1)^i \leq (l+1)^{l+1} \leq 2^{1024} \tag{18}$$

The biggest value of l that satisfies (18) is 193. This technique can trace the product in supply chain contains up to 193 steps.

8.2 Storage cost

The storage cost is computed as the number of bytes that the sensor node has to store. Generally, this storage cost is introduced by the storage of different parameters and keys necessary to the function of our scheme. The proposed privacy preserving product tracking scheme does not require much memory overhead.

- Initialization phase: in this phase, the sensor stores Rabin's public key of size $sizeof(N_R)$, the path trace initialization of size $sizeof(\mathcal{E}(T))$, paillier modular N^2 of size $2 * sizeof(N)$=$sizeof(\mathcal{E}(T))$ and a sensor ID of size $sizeof(S_{id})$. Therefore, the total storage needed by S at this phase is $2 * sizeof(\mathcal{E}(T))$, + $sizeof(S_{id})$ + $sizeof(N_R)$.
- Collection phase: in this phase, S has to generate random number to start the authentication with the actor's system. The generated nonce has a size of $sizeof(N_R)$. Hash value of the generated nonce of size $sizeof(hash)$ has to be stored as well. Therefore, the total storage needed by S at this phase is $sizeof(N_R)$ + $sizeof(hash)$. The update path process does not increase the size of the path trace, thus, no more memory capacity is required.
- Verification phase: in this phase, no storage by the sensor is required.

The total storage cost needed by S in our scheme is:

$$\begin{aligned} storageCost =& 2 * sizeof(\mathcal{E}(T)) + sizeof(S_{id}) \\ &+ 2 * sizeof(N_R) + sizeof(hash) \end{aligned} \tag{19}$$

Table 6 illustrates the storage cost for our scheme.

Table 6. storage cost of SPT scheme

Parameter	value
sizeof(S_{id})	160 bits
sizeof(N_R)	1024 bits
sizeof($hash$)	160 bits
sizeof($\mathcal{E}(T)$)	2048 bits
The storage cost	6378 bits

8.3 Computation cost

The computation cost can be measured in terms of time, use of CPU or energy dissipation. In fact, these parameters are related and each one can be deduced from the other. For instance, the energy dissipation can be deduced from the time as follows: Energy=Power*Time, where Power represents the CPU power when it is in its active state and Time represents the computing time. In the present analysis, the term cost is used in its general form without specifying the unit. The computation cost of our scheme during each phase can be computed as the sum of the computation cost of the main operations executed during this phase.

- Initialization phase: in this phase, the main operations are performed by M himself. Therefore, no computation required by the sensor S.
- Collection phase: in this phase, S generates a random number $rand$ at each step, which gets encrypted using Rabin scheme. Then, S computes a hash function of $rand$ in order to check the validity of the actor's response. If the validation is succeeded, S updates the path trace by modular multiplying the received step's ID with the current trace. In total, S consumes $cost(rabin)$ + $cost(rand)$ + $c(hash)$ +$cost(modularmultiplication)$ for each step.
- Verification phase: in this phase, S has to authenticate the supply chain manager M, which is similar to authenticate a supply chain actor. Therefore, S consumes in this phase, $cost(rabin)$ + $cost(rand)$ + $cost(hash)$.

The total computation cost needed by S in our scheme is:

$$\begin{aligned} compCost =& (cost(rabin) + costrand + costhash) * (l + 1) \\ &+ cost(modularmultiplication) * l \end{aligned} \tag{20}$$

where l is the number of the supply chain steps that the product has visited. Table 7 illustrates the computation cost for our scheme.

Table 7. computation cost of SPT scheme

Parameter	value
$cost(rabin)$	$100\mu J$
$cost(rand)$	$58\mu J$
$cost(hash)$	$1\mu J$
$cost(modularmultiplication)$	$100\mu J$
The computation cost	$160,59mJ$ for supply chain with 100 steps

8.4 Communication cost

The main factor of the communication cost is the energy dissipation. The communication cost is computed using the same approach as TKH [16]. Actually, the communication cost in terms of energy dissipation is computed as the size of sent/received messages multiplied by the energy dissipated for the sent/receive of one bit. We denote e_r the energy consumed by S, when it receives one bit, and e_s when S sends one bit.

- Initialization phase: in this phase, S does not send any messages, however, it receives the initialization parameters. Therefore, S consumes (sizeof(N_R)+ sizeof(S_{id})+2*sizeof($\mathcal{E}(T)$))*e_r.
- Collection phase: in this phase, S sends encrypted rabin value, which has the same size as the Rabin's public key, and it receives a hashed value. Then, S receives the current encrypted step's ID from the actor's site. Therefore, S consumes (sizeof(N_R)*e_s+ (sizeof($hash$) +sizeof($\mathcal{E}(T)$))* e_r for each step.
- Verification phase: in this phase, S authenticates M, then, sends the path trace to him. Therefore, S consumes (sizeof(N_R)+ sizeof(S_{id}+ sizeof($\mathcal{E}(T)$))*e_s+ (sizeof($hash$))* e_r

The total communication cost needed by S in our scheme is:

$$\begin{aligned} commCost =& (sizeof(N_R) + sizeof(hash) * (l+1) \\ &+ sizeof(S_{id}) + sizeof(\mathcal{E}(T)) * (l+2)) * e_r \\ &+ (sizeof(N_R) * (l+1) + sizeof(\mathcal{E}(T))) * e_s \end{aligned} \tag{21}$$

where l is the number of the supply chain actors that interact with the product. Table 8 illustrates the communication cost for our scheme.

Table 8. communication cost of SPT scheme

Parameter	value
sizeof(S_{id}	160 bits
sizeof(N_R)	1024 bits
sizeof($hash$)	160 bits
sizeof($\mathcal{E}(T)$)	2048 bits
e_r	$0,209\mu J$
e_s	$0.226\mu J$
The communication cost	$12,34mJ$ for supply chain with 100 steps

It is worth to mention that the storage cost, computation cost, and the communication cost can have a different result depending on the size of the security keys, and the algorithms that M may choose.

9 Conclusion

In this paper, we presented a cryptographic protocol to address security and privacy challenges in tracking the goods within supply chain management. Our main idea is to use AND-ACC code as identifiers of the different actors in the supply chain. Then, use a mathematical transformation to convert their AND property to additive property. Finally, an additive homomorphic encryption is used to ensure both the confidentiality of the identities, and the compression in the sensors.
Sensors collect the identities of the visited steps , and update their path traces. This allows supply chain manager to trace back all the steps that a product went through. The security of our protocol against adversaries relies on the semantic security of Paillier and the indistiguishibility property of keyed HMAC. Rabin's scheme used to check the legitimacy of the supply chain actors, however, other techniques can be used such as the one presented by Sorniotti et al. [17].
In our supply chain scenario, we assume that we have a global supply chain manager. There is no notion of multiple managers. However in real world that might not be true. Supply chain can have a quality, security, and recall mangers. Delivering the right information to the right manager is an issue, especially in big scale supply chains. However, this is left to future work. [3]

References

1. E. Blass, K. Elkhiyaoui, and R. Molva. Tracker : security and privacy for rfid-based supply chains. In *NDSS'11, 18th Annual Network and Distributed System Security Symposium, 6-9 February 2011, San Diego, California, USA, ISBN 1-891562-32-0*, 02 2011.
2. K. Chawla, G. Robins, and W. Weimer. On Mitigating Covert Channels in RFID-Enabled Supply Chains. *RFIDSec Asia, Singapore*, 2010.
3. A. Francillon and C. Castelluccia. Tinyrng: A cryptographic random number generator for wireless sensors network nodes. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, 2007. WiOpt 2007. 5th International Symposium on*, pages 1–7. IEEE, 2007.
4. G. Gaubatz, J.P. Kaps, and B. Sunar. Public key cryptography in sensor networksrevisited. *Security in Ad-hoc and Sensor Networks*, pages 2–18, 2005.
5. R. Gupta and S.R. Das. Tracking moving targets in a smart sensor network. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, pages 3035–3039. IEEE, 2003.
6. M. Hempstead, M.J. Lyons, D. Brooks, and G.Y. Wei. Survey of hardware systems for wireless sensor networks. *Journal of Low Power Electronics*, pages 11–20, 2008.
7. http://www.atmel.com/Images/doc2467.pdf. Last access: 01/06/2012.
8. http://www.xbow.com/. Last access: 01/06/2012.

[3] research was partially funded by the German Federal Ministry of Education and Research under the promotional reference 01ISO7009 and by the French Ministry of Research within the RESCUE-IT project. The authors take the responsibility for the content

9. F. Kerschbaum and R.J. Deitos. Security against the business partner. In *Proceedings of the 2008 ACM workshop on Secure web services*, pages 1–10. ACM, 2008.
10. S.J. Lee and S.H. Jung. A survey of watermarking techniques applied to multimedia. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 1, pages 272–277. IEEE, 2001.
11. K. Ouafi and S. Vaudenay. Pathchecker: An RFID Application for Tracing Products in Supply-Chains. In *International Conference on RFID Security*. Citeseer, 2009.
12. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology, EUROCRYPT99*, pages 223–238. Springer, 1999.
13. K.R. Pattipati, S. Deb, Y. Bar-Shalom, and R.B. Washburn Jr. A new relaxation algorithm and passive sensor data association. *Automatic Control, IEEE Transactions on*, pages 198–213, 1992.
14. M.O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. 1979.
15. H. ShuiHua and C.H. Chu. Tamper Detection in RFID-Enabled Supply Chains Using Fragile Watermarking. In *RFID, 2008 IEEE International Conference on*, pages 111–117. IEEE.
16. Ju-Hyung Son, Jun-Sik Lee, and Seung-Woo Seo. Topological key hierarchy for energy-efficient group key management in wireless sensor networks. *Wirel. Pers. Commun.*, 52(2):359–382, January 2010.
17. A. Sorniotti, R. Molva, and L. Gomez. Efficient access control for wireless sensor data. *Ad Hoc & Sensor Wireless Networks*, pages 325–336, 2009.
18. TAGSYS. Rfid luxury goods solutions. 2012.
19. W. Trappe, M. Wu, Z.J. Wang, and K.J.R. Liu. Anti-collusion fingerprinting for multimedia. *Signal Processing, IEEE Transactions on*, 51(4):1069–1087, 2003.
20. VERAYO. unclonable-rfids. 2012.