Distributed Path Authentication for Dynamic RFID-Enabled Supply Chains

Shaoying Cai, Yingjiu Li, Yunlei Zhao

Distributed Path Authentication for Dynamic RFID-Enabled Supply Chains*

Shaoying Cai[1], Yingjiu Li[1], Yunlei Zhao[2]

[1]Singapore Management University, 80 Stamford Road, Singapore 178902
[2]Fudan University, No. 825 Zhangheng Road, Shanghai, China 201203
shaoyingcai.2009@smu.edu.sg, yjli@smu.edu.sg, ylzhao@fudan.edu.cn

Abstract. In this paper, we propose a distributed path authentication solution for dynamic RFID-enabled supply chains to address the counterfeiting problem. Compared to existing general anti-counterfeiting solutions, our solution requires non sharing of item-level RFID information among supply chain parties, thus eliminating the requirement on high network bandwidth and fine-grained access control. Our solution is secure, privacy-preserving, and practical. It leverages on the standard EPCglobal network to share information about paths and parties in path authentication. Our solution can be implemented on standard EPC class 1 generation 2 tags with only 720 bits storage and no computational capability.

1 Introduction

Supply chain is a network involving multiple parties such as suppliers, transporters, storage facilities, distributors, and retailers that participate in the production, delivery, and sale of a product [5]. It is difficult to monitor a supply chain since the involving parties are distributed at multiple locations or even across countries. It is therefore critical to address the counterfeiting problem, where an adversary injects fake goods into a supply chain. The counterfeiting problem has become a major threat to supply chains. According to the 2011 report of International Chamber of Commerce, it is estimated that the counterfeiting accounts for 5-7% of world trade, or about 600 billion U.S. dollars per year [6]. The ratio of counterfeiting is even higher in dynamic supply chains where the involving parties and the paths of processed goods may not be fixed.

Radio Frequency IDentification (RFID) technology has been recently used to facilitate real-time monitoring of supply chains so as to thwart counterfeiting threats. Most existing solutions in industry rely on sharing and processing massive RFID information at item level collected by each supply chain party over the internet. Such solutions inevitably incur high network bandwidth (due to large volume of processed goods) and fine-grained access control (due to security requirements by different parties). It is therefore difficult, even impossible,

* The second author's work is supported in part by the Office of Research at Singapore Management University. The third author is supported in part by an NSFC grant No. 61070248, and a grant from Shanghai Municipal Education Commission.

to implement such solutions in dynamic supply chain environments, where the involving parties may not have pre-existing trust relationship for sharing their RFID information in a secure and efficient manner.

Since dynamic supply chains are prevalent in living supply chains [16], we propose a new distributed path authentication scheme to thwart counterfeiting in such environment. Our scheme verifies whether a tag comes from a reliable source and has been precessed by a series of legitimate entities in a supply chain. Compare to existing solutions, our scheme eliminates the needs of sharing item-level information among supply chain parties; thus, it does not require pre-existing trust relationship nor fine-grained access control. The verification of a tag's path is based on the information carried by the tag and certain auxiliary information obtained from a trusted server in EPCgloabel network, which is a standard infrastructure for RFID-enabled supply chains. While our scheme is designed for the most dynamic RFID-enabled supply chains, it is also suitable for a supply chain that has fix partnership or/and static goods processing procedure.

The challenges of designing our scheme come in two aspects. First, the tag storage is restricted to no more than 1088 bits for the most popular low-cost standard C1 G2 tags [4]. In order to prove that a tag has been processed by a series of entities, the tag should carry certain credentials generated by the entities. To avoid complicated key management at item level, a natural way is to use the entities' signatures on a tag's ID as the credentials. However, it is not practical to store all signatures and public keys on a tag as the tag's storage is limited, especially in the case that such information may continually increase as the tag goes through more entities. Our solution keeps the information stored on a tag in constant size, which is less than 1088 bits, satisfying the storage constraint for EPC C1 G2 standard tag. In our solution, an ordered multi-signature scheme [11] is adopted to generate a constant-size signature of the entities on the tag's ID. A path index is used to indicate the series of entities which have processed a tag. The index is stored on the tag instead of the entities' public keys, while the detailed information about the path is stored on a trusted server such as EPCglobal Discovery Server.

The second challenge is to reduce the communication load between an entity and the trusted server when verifying a signature. To verify a signature, an entity queries the trusted server with the index of the path. The server sends the public keys of the entities in the path to the entity. The communication load increases as the path getting longer in a naive solution. We reduce the communication load to a constant level regardless of path length in our solution. Due to the specific constructions of the underlying ordered multi-signature scheme, in our scheme, the server only needs to send an aggregated "path public key" instead of the public keys of the entities. With the "path public key", one can verify whether a signature is generated orderly by the entities in the path or not. The "path public key" even has smaller size than a single public key. In the case where a batch of tags share the same path, a verifier only needs to query the trusted server once for the aggregated "path public key" in practice.

The security of our scheme relies on the unforgeability of the underlying ordered multi-signature scheme. Our scheme is secure in a sense that an adversary cannot forge any valid tag or path. To protect the privacy of each tag, we take advantage of supply chain's batch processing property. In a batch, each tag's information is encrypted with the same key. The key is divided to several shares with a secret sharing scheme. Each tag stores a share of the key together with its encrypted information. Only authorized readers can access the whole batch of tags, recover the key and then decrypt the contents stored on the tags. Our scheme is privacy preserving in a sense that an adversary cannot identify any valid tag, or distinguish whether two valid tags have taken the same path or not. Our scheme leverages on standard EPCglobal network and can be implemented on standard EPC class 1 generation 2 tags with only 720 bits storage and no computational capability.

2 Background

Many protocols such as [23,29] have been designed to provide mutual authentication or tag authentication in RFID systems. These protocols can be used to prevent counterfeiting in supply chains. However, in order to monitor a supply chain, these proposals require the manager to access the databases of all the entities in the supply chain. Therefore, they rely on high quality network connection and fine-grained access control. These requirements are obstacles in deployment of dynamic supply chain management systems. In addition, most of the existing solutions rely on non-standard tags with certain computational abilities, such as hash operation. This will incur high cost for not using standard low-cost tags.

The most related work address the counterfeiting problem in RFID-enabled supply chains based on standard EPC class 1 Gen 2 tags. In Zanetti, Fellmann and Capkun's scheme [30], the entities cooperate together to verify the tags' genuineness without revealing information to each other. This scheme cannot resist tracking attack since any reader can read out each tag's ID. Blass, Elkhiyaoui and Molva proposed TRACKER [10] and its extension [9]. In TRACKER system, each tag stores encrypted verifiable ID and path information. The manager decrypts the information and verifies whether the tag has gone through a valid path. TRACKER dose not need the entities in a supply chain to have any connection except the initialization phase. However, it requires the manager to possess the secret keys of all involving entities, the manager can only verify the tags from a set of pre-fixed paths. Therefore, it is difficult to implement TRACKER in dynamic supply chains.

2.1 Dynamic RFID-enabled supply chain

A supply chain consists of multiple entities. We model a dynamic supply chain according to three properties: the affiliation of each entity, the membership management of the supply chain and the logistics flow of the supply chain. In a dynamic supply chain, each entity is independent of each other; any entity can

freely join or leave; the logistics flow is not fixed. An RFID-enabled dynamic supply chain management system should meet the requirements below.

- Each reader is independent with other readers and has an unique ID. Note that some readers may belong to a same entity. Even that, we assume the readers are independent with each other and have no pre-existing connections, such as network connection among themselves to cater for the most flexible deployment condition.
- Each reader does not share its secret (eg. private key) with other readers.
- Each reader in the supply chain is isomorphic, with the same functionalities. Each reader should be able to initialize the tags, identify the tags, verify the tags and update the tags.

2.2 Adversary model

Figure 1 illustrates our adversary model. The entities in a supply chain provide relative secure environments within their teritories, where the tags are beyond the accessible distance of an adversary. During the transportation of tagged goods between entities, a "hit and run" adversary can only approach the goods for a short period of time from a not-so-close distance.

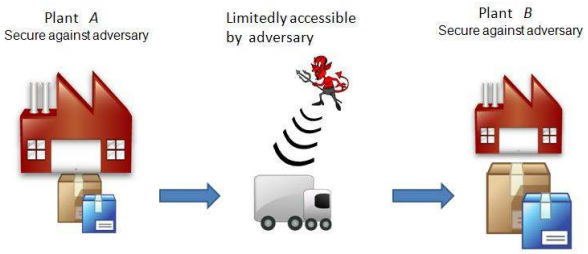

Fig. 1. Adversary Model of Supply Chain

The "hit and run" adversary modle was firstly proposed by Ari Juels [18]. Several works [22,21,19] have also adopted this model in designing secure RFID systems. The rationals to adopt this model in supply chains are: 1) the tagged goods usually rapid change their physical locations and ownerships so that it is difficult for an adversary to keep in the working distance of the tags (normally no more than ten meters); 2) as both readers and tags work in short range, an adversary bringing a reader into a monitored environment like a shop or warehouse might face difficulties in attempting prolonged intelligence gathering [18].

2.3 Requirements of RFID-enabled path authentication system

In dynamic supply chains, we list the following practice requirements for designing RFID-enabled path authentication:

– Any valid reader can extract a tag's ID and the exact path that the tag has passed through in the supply chain. However, no readers needs to store the information of all possible paths in its own database in advance.
– An adversary cannot create new tag or modify existing one without being detected. A tag cannot pass the verification process for a path by which it has not passed.
– No efficient adversary can link the state information stored in a tag to the tag's identity. No efficient adversary can distinguish whether two tags have taken the same path or not.
– Path authentication can be performed on general EPC Class 1 Gen 2 tags, which have at most 1088 bits and no computational capability.

3 Our construction

Our system contains three components: tags, readers and a trusted server. Each tag stores a tag ID, a path code, and a signature on the tag's ID generated by the readers in the path. Any valid reader has a pair of public key and private key and a random number used for generating the path code. A valid reader is able to extract each tag's ID and the path code; while to verify them, it needs to connect with the trusted server which stores the reader's public information and detailed path information. In designing a secure and privacy-preserving path authentication scheme, we use the following building tools.

3.1 Building tools

Bilinear map: A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to G_T$, where: (a) $\mathbb{G}$ is a (multiplicative) cyclic groups of prime order p; (b) $|\mathbb{G}| = |\mathbb{G}_T|$; (c) g is a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the following properties: (a) Bilinear: for all $x, y \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$; (b) Non-degenerate: $e(g, g) \neq 1$. We call an algorithm $\mathcal{G}$ that outputs $(p, \mathbb{G}, \mathbb{G}_T, e)$ as above a bilinear-group generation algorithm.

Path encoding method: Noubir et al. [25] proposes to encode a software's state machine using polynomials such that the exact sequence of states visited during run-time generates a unique "mark". We adopt this technique in generating the path code. Suppose there is a path $P_v = \{R_{v_1}, \cdots, R_{v_{l_v}}\}$, where l_v is the length of path P_v, v_i represents the reader's identity of the ith step in path P_v; we assign each reader R_j with a unique random number $a_j \in \mathbb{F}_q$, where q is a large prime. A path is represented with a polynomial on $\mathbb{F}q$. Then the polynomial corresponding to a path $P_v = \overrightarrow{R_{v_1} \cdots R_{v_l}}$ is defined below (all operations are in $\mathbb{F}q$.):

$$Q_{P(x)} := \sum_{i=1}^{l} a_{v_i} x^{l-i} \tag{1}$$

Given a generator x_0 of $\mathbb{F}_q$, we calculate the path code as $\phi(P_v) := Q_{P_v}(x_0)$ and identify a path P_v using its polynomial evaluation $\phi(P_v)$.

Secret Sharing Scheme A (τ, n)-secret sharing scheme is an algorithm that divides data D into n pieces in such a way that: 1) knowledge of any τ or more pieces makes D easily computable; 2) knowledge of any $\tau - 1$ or fewer pieces leaves D completely undetermined (in the sense that all its possible values are equally likely) [28]. We adopt the Tiny Secret Sharing (TSS) [19] proposed by Juels et al. in our construction.

Ordered Multisignature Scheme Ordered multisignature scheme (OMS) allows signers to attest to a common message as well as the order in which they signed. The concept is raised by Boldyreva et al. in [11]. A construction of OMS is provided in [11], and we denote it as BGOY-OMS scheme from now on. We summarize BGOY-OMS system as follows. Each BGOY-OMS system has global information $I = (p, \mathbb{G}, \mathbb{G}_T, e, g, H)$, where $(p, \mathbb{G}, \mathbb{G}_T, e)$ is generated by a bilinear-group generation algorithm $\mathcal{G}$, g is a random generator of $\mathbb{G}$, and $H : \{0,1\}^* \rightarrow \mathbb{G}$ is cryptographic hash function.

- **Key Generation:** On input I, the algorithm chooses random $s, t, u \in \mathbb{Z}_p$ and returns $(S = g^s, T = g^t, U = g^u)$ as pk and (s, t, u) as sk.
- **Signing:** On inputs sk_i, m, σ, $L = (pk_1, \cdots, pk_{i-1})$, the algorithm first verifies whether Equation 2 defined below holds and if not, outputs $\perp$. (This step is skipped for the first signer, i.e. if $i = 1$, for whom σ is defined as $(1_G, 1_G)$.) Then it parses σ as (Q, W) and chooses random $w \in \mathbb{Z}_p$ and computes $W' = W \cdot g^w$, $X = (W')^{t_i + iu_i}$, $Y = (\prod_{j=1}^{i-1} T_j (U_j)^j)^w$ and $Q' = H(m)^{s_i} \cdot Q \cdot X \cdot Y$. Finally, it returns (Q', W').
- **Verification:** On inputs $\{(pk_1, \cdots, pk_n), m, \sigma\}$, the algorithm first checks that all of $pk_1, \cdots, pk_n$ are distinct and outputs 0 if not. Then it parses σ as (Q, W) and checks if

$$e(Q, g) \stackrel{?}{=} e(H(m), \prod_{i=1}^{n} S_i) \cdot e(\prod_{i=1}^{n} T_i (U_i)^i, W). \tag{2}$$

  If so, it outputs 1. If not, it outputs 0.

3.2 Protocol details

The tag information, including tag ID, path code, and signature is encrypted. The tags in the same batch share the same encryption key. Then the encryption key is distributed using TSS secret sharing scheme and each tag stores a share of the key together with encrypted data. A valid reader can collect enough shares, recover the key, and decrypts the information of each tag. For each tag, after decryption, a valid reader obtains the tag ID, a path code, and a signature on the tag ID. Querying a trusted server with the path code, the reader gets the aggregated "path public key" of the path which is computed from the readers' public keys, and uses it to verify the signature. If the tag is valid, then the reader can update the path code and the signature and encrypt them with a new random key. Finally, the reader stores the new tag information on the tag. The tags are

processed by batch in supply chain management. Polynomial signature based path encoding method [25], BGOY-OMS [11], and TSS [19] are incorporated in the design of our system.

System Setup: A BGOY-OMS system is set up by running a bilinear-group generation algorithm $\mathcal{G}$ for output $(p, \mathbb{G}, \mathbb{G}_T, e, g, H)$. Choosing a large prime q, and a random number $x_0 \in \mathbb{Z}_q$, we get $I = (p, q, \mathbb{G}, \mathbb{G}_T, e, g, H, x_0)$, which is the global information for the scheme. Assume that there are m readers in total. Each reader R_j is assigned with public key $pk_j = (S_j, T_j, U_j)$ and secret key $sk_j = (s_j, t_j, u_j)$, where s_j, t_j, u_j are randomly chosen from $\mathbb{Z}_p$, $1 \leq j \leq m$. Each reader R_j is also assigned with a random number $a_j \in \mathbb{Z}_q$, where a_j will be used in generating a path code.

A trusted server publishes the global information I, each reader's public key pk_j and random number a_j. The trusted server also stores each path P_v 's information, including "$pathcode_v$, l_v, $P_v = \{R_{v_1}, R_{v_2} \cdots, R_{v_{l_v}}\}$, $ppk_v = (ppk1_v, ppk2_v) = (\prod_{j=1}^{l_v} S_{v_j}, \prod_{j=1}^{l_v} T_{v_j}(U_{v_j})^j)$", where l_v is the number of readers in path P_v, v_j denotes the identity of the jth reader in path P_v, $pathcode_v$ is the path code of P_v generated using Equation (1) in $\mathbb{F}_q$, and $ppk_v = (ppk1_v, ppk2_v)$ $= (\prod_{j=1}^{l_v} S_{v_j}, \prod_{j=1}^{l_v} T_{v_j}(U_{v_j})^j)$ is each path's public key stored in a path record. With ppk_v, a valid reader can verify the signature on the tag without knowing any other reader's individual public key. This will reduce the communication load between a reader and the trust server. In case a reader needs to verify the signature on a tag based on all involving readers' public keys, it can also get the public keys from the trusted sever. Table 1 shows the contents stored on the trusted server.

Table 1. Contents on Trusted Server

System Parameters	$(p, q, \mathbb{G}, \mathbb{G}_T, e, g, H, x_0)$
Reader Information R_j, for $1 < j < m$ m is the number of readers	$pk_j = (S_j, T_j, U_j), a_j$
Path Information P_v	$pathcode_v$, l_v, $P_v = \{R_{v_1}, R_{v_2} \cdots, R_{v_{l_v}}\}$, $ppk_v = (ppk1_v, ppk2_v) = (\prod_{j=1}^{l_v} S_{v_j}, \prod_{j=1}^{l_v} T_{v_j}(U_{v_j})^j)$

Batch initialization of the tags: Suppose that a batch $\mathcal{T}$ of n tags enters in a supply chain, where each tag is denoted as T_i with unique ID id_i, for $i \in \{1, \cdots, n\}$. The tags can be initialized by a reader valid R_x, $1 \leq x \leq m$. In particular, R_x generates a key k and n shares of k using TSS-scheme, where each share is denoted as s_i, for $1 \leq i \leq n$. For each tag T_i, R_x generates a signature $\sigma_i = (Q_i, W_i)$ on the tag ID id_i under its private key using BGOY-MOS scheme. Then R_x sets $pathcode_i$ of each tag T_i to a_x. Finally, R_x encrypts $(id_i, \sigma_i, pathcode_i)$ with the key k, and stores $\{s_i, E_k(id_i, \sigma_i, pathcode_i)\}$ on T_i. After initializing the batch of tags, R_x queries the trust server to check whether the path $P = \{a_x, 1, \{R_x\}, ppk = (S_x, T_x U_x)\}$ already exists; if not, R_x inserts

path P to the database on the trust server. Then R_x releases the batch of tags into the supply chain.

Interactions between reader and tag: When a batch $\mathcal{T}$ of tags in the supply chain arrives at reader R_y, where $1 \leq y \leq m$. Each tag T_i in the batch stores a state $st_i = \{s_i, E_k(id_i, \sigma_i = (Q_i, W_i), pathcode_i)\}$. Reader R_y firstly reads all the tags in $\mathcal{T}$ to get st_i for all $1 \leq i \leq n$. Using at least τ shares, R_y recovers a key k, and decrypts the information on each tag $E_k(id_i, \sigma_i, pathcode_i)$ and gets $\{id_i, \sigma_i = (Q_i, R_i), pathcode_i\}$. According to $pathcode_i$, R_y gets the path's information $P = \{pathcode_i, l, \{R_{P_0}, \cdots, R_{P_l}\}, ppk = (ppk_1, ppk_2)\}$ form the trusted server. If $e(Q_i, g) = e(H(id_i), ppk_1) \cdot e(ppk_2, W)$, then T_i passes the verification. To update the batch of tags, R_y generates a new key k' and n shares of k' in TSS, where each share is denoted as s'_i. For each tag T_i, R_y shall update both the signature σ_i and the path code $pathcode_i$. To do these, R_y chooses a random number w in $\mathbb{Z}_p$, computes $W'_i = W_i \cdot g^w$, $Q'_i = W_i'^{t_y + (l+1)u_y} \cdot ppk_1^w \cdot Q \cdot H(id_i)^{s_y}$ and $pathcode'_i = pathcode_i \cdot x_0 + a_y$. R_y updates each tag T_i by writing $\{s'_i, E_{k'}(id_i, \sigma'_i = (Q'_i, W'_i), pathcode'_i)\}$ to the tag. Finally, the reader queries the trusted server: if the path $P_{new} = \{pathcode'_i, l+1, \{R_{P_0}, \cdots, R_{P_l}, R_y\}, ppk = (ppk_1 \cdot S_y, ppk_2 \cdot T_y U_y^{l+1})\}$ does not exist in the trusted server, then path P_{new} will be added for future queries.

Implementation details: TSS scheme can be implemented with Reed-Solomon code [26]. A Reed-Solomon code is specified as $RS(N, K)_S$. A codeword has S bits. A reader chooses a pre-key with $K \cdot S$ bits and encodes the pre-key to N shares with each share S bits. A hash value of the pre-key is used as the encryption key for a batch of N tags. Reed Solomon encoding and decoding have been implemented on an Intel Core 2 CPU 6320 1.86GHz in [2]. Choosing the parameters (N, K, S) as $(32768, 16384, 16)$, the fastest algorithm achieves 6.17 Mbytes throughput per second for encoding, and 2.73 Mbytes throughput per second for decoding. Both encoding procedure and decoding procedure consume less than one second. Suppose there are 20000 tags in a batch, one can firstly choose a pre-key with $16 * 16384$ bits, encode them to 32768 symbols. Each share contains one symbol. Then one can select 20000 shares, randomly store on share on each tag. Any reader that can successfully read more than 16384 tags is able to get the pre-key. Regarding the encryption algorithm, Blowfish [3] is an appropriate choice. According to [3], Blowfish has good performance that achieves 64.386MB per second of encryption throughput on a Pentium 4 2.1 GHz processor under Windows XP SP 1.

4 Analysis

Both the security and privacy of our scheme rely on the security and privacy properties of the underlying secret sharing scheme, encryption scheme and OMS scheme. Due to space limit, we leave detailed proof to an extended version of this paper.

4.1 Security

We assume that in supply chain management, an adversary's goal is to insert counterfeited goods into the supply chain. The security goal of our system is

to prevent an adversary from forging a tag's internal state so that the tag is considered from a reliable source and has gone through a valid path that has not actually been taken by the tag in the supply chain.

The security of our solution is based on the unforgeability of the BGOY-OMS scheme. Intuitively, the unforgeability of BGOY-OMS scheme can be described as follows: given an uncorrupted party with key pair (pk, sk), a forger cannot generate a valid BGOY-OMS signature of the uncorrupted party on any message m' if the forger does not know the party's signature on m'. Note that in the original BGOY-OMS unforgeability game, given a key pair (pk, sk), the adversary is able to get signature on any tag's ID under the key sk so as to learn useful information. In our system, the adversary cannot get a valid reader's signature on any ID it chooses since the valid reader will verify the genuineness of any tag before generating a signature on its ID. Hence, the adversary in our system is weaker than the adversary in the BGOY-OMS unforgeability game. BGOY-OMS scheme is secure against the forger; hence it is also secure against the adversary in our system. Unless an adversary has corrupted all the readers in a path with path code *pathcode*, it cannot forge a tuple $(ID, \sigma,$ *pathcode*$)$ such that σ is a valid signature on ID generated by the readers in the path.

While an adversary cannot forge any valid new tuple $(ID, \sigma,$ *pathcode*$)$ by itself, another way to counterfeit an tag is to use an existing valid tuple in the supply chain. From this aspect, the counterfeiting countermeasure of our system relies on the batch processing property. Only can valid readers read the whole batch of tags and decrypt the contents of the tags. A "hit and run" adversary is not able to read more than $\tau - 1$ tags in a batch , thus cannot recover the decryption key. In EPC Class 1 Gen 2 tags, the user memory bank can be protected by access pin. We use the encryption/decryption key as the access pin for the tags; therefore, an adversary cannot get the complete content stored on a tag and our system is secure against the counterfeiting attack.

4.2 Privacy

The privacy of RFID path authentication can be considered at two levels: tag unlinkability and path unlinkability. Tag unlinkability requires that no efficient adversary can link the state information stored in a tag to the tag's identity. Path unlinkability requires that no efficient adversary can distinguish whether two tags have taken the same path or not. In our scheme, each tag stores a copy of encrypted ID, pathcode and signature together with a single share of the encryption key. The privacy of our system relies on honest behaving of valid readers. Each valid reader uses a new random key to encrypt the updated state for each tag. An hit-and-run adversary who cannot collect enough shares for recovering the encryption keys cannot distinguish between any two ciphertexts of the same tag and any two ciphertexts of different tags. Thus our scheme preserves tag unlinkability. Similarly, our scheme preserves path unlinkability since an adversary cannot obtain any information about a tag's path from the content of the tag.

4.3 Performance

We analyze the performance of our solution in three aspects: computation requirements, communication requirements and storage requirements.

Computational requirements: Our scheme does not require tag to perform any computation. All the computation can be performed at RFID reader side. In computing the computational load of a reader, we omit the cheap operations such as Reed Solomon encoding, decoding, encryption and decryption using Blowfish, hash operation, and point addition on elliptic curve. We only count the relative expensive operations such as point multiplication on elliptic curve and paring. A reader needs to perform three paring operations to verify the signature and four point multiplication four updating the signature in each tag.

Due to batch processing in supply chain, we can reduce the computational cost if a batch of tags share the same path. Assuming that a batch of tagged goods is transferred in a supply chain without being mixed with other goods, then a reader can sign the batch of tags with the same random number. That is, each tag shares the same randomization factor W in the signature. To update the signature, the reader firstly computes (W', X, Y), which requires three point multiplication operations, and uses (W', X, Y) to generate each tag's signature. With (W', X, Y), the reader's computational load is reduced to one point multiplication in generating each tag's signature. For signature verification, since each tag in the batch stores the same randomization factor W in the signature, a reader can pre-compute $e(ppk_2, W)$ and store the value. The computational requirements for each tag is reduced to two paring operations. Since all the computation can be performed on reader side, our solution is applicable on standard low-cost tags with no computation capability.

Communication requirement: The verification of each tag's ID and path code requires a valid reader to connect to the trusted server. The reader sends the path code to the server, then the server returns necessary path's information. In the case that a batch of tags passed the same path, the reader needs only one path information from the server. Since batch processing is commonly used in supply chain practice, the communication load required for processing a batch of tags should be almost constant. The communication load can be further reduced if a reader stores path information for frequently used paths in its own database.

Storage requirement: In each tag T_i, we need to store $\{s_i,\ E_k(id_i,\ \sigma_i = (Q_i, W_i), pathcode_i)\}$. s_i is a share of encryption key k. As we implement TSS scheme with Reed-Solomon code, s_i can be a symbol, which we use 16-bit string (a symbol's length depends on the parameters of Reed-Solomon code). Tag ID id_i is an EPC code, which has 96 bits. Tag signature σ_i generated by BGOY-OMS scheme consists of two elements on $\mathbb{G}$. For 80-bit security level with embedding degree $k = 2$, an element in $\mathbb{G}$ can be represented in 512 bits. For embedding degree $k = 6$, the length can be reduced to 237 bits [13]. Hence, the storage requirement for σ_i is at least 474 bits. We use 80 bits to represent a path ID, which supports at most 2^{80} different path codes. We adopt Blowfish block cipher [27] for encryption which has a block size of 64-bits. Thus, we need 720 bits in total. Our system can thus be implemented with EPC Class 1 Gen 2 tags with

an extensible EPC memory bank (scalable between 16-480 bits), a scalable user memory bank (64-512 bits), a TID bank (32 bits), and a reserved bank (64 bits), which are available on the market [4].

5 Conclusions

In this paper, we proposed a distributed path authentication scheme for dynamic supply chains. Dynamic supply chain is a prevalent supply chain structure in real world that is inherently vulnerable to the injection of counterfeiting goods and at same time challengeable to design an RFID-enabled system which can monitor the whole supply chain. We design a path authentication scheme to enable any valid reader to verify the exact path that a tag has taken without requiring the reader to have any kind of connection with other readers; it is thus suitable for dynamic supply chains where supply chain partners (or readers) may not have pre-existing trust relationship. Our scheme is built upon tiny secret sharing scheme, multisignature scheme, polynomial signature scheme, and encryption scheme. Our system is secure, privacy-preserving and practical. Our scheme leverages on sharing path information on standard EPCglobal network, and can be implemented on standard low-cost RFID tags with no computation capability and limited memory.

References

1. http://www.edn.com/article/458737-Counterfeitcomponentsremains_a_huge_electronics_supply_chain_problem.php.
2. http://algo.epfl.ch/~didier/reed_solomon.html.
3. http://www1.cse.wustl.edu/~jain/cse567-06/ftp/encryption_perf/index.html.
4. http://www.alientechnology.com/tags/index.php.
5. http://www.wisegeek.com/what-is-a-supply- chain.htm.
6. ICC Commercial Crime Services. Counterfeiting intelligence bureau. 2011. http://www.icc-ccs.org/home/cib.
7. C. *Ó* h*É*igeartaigh. Pairing computation on hyperelliptic curves of genus 2. *PhD thesis*, Dublin City University, 2006.
8. G. Ateniese, J. Camenisch, and B. D. Medeiros. Untraceable RFID tags via insubvertible encryption. In *CCS*, pages 92–101, New York, USA, 2005.
9. E. O. Blass, K. Elkhiyaoui, and R. Molva. Tracker: Security and privacy for RFID-based supply chains. *Cryptology ePrint Archive*, Report 2010/219, 2010.
10. E. O. Blass, K. Elkhiyaoui, and R. Molva. Tracker : security and privacy for RFID-based supply chains. In *NDSS*, San Diego, California, USA, 2011.
11. A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *CCS*, pages 276–285, New York, NY, USA, 2007.
12. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. In *CryptoBytes*, Vol. 6, No. 2, 2003.
13. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, London, UK, 2001.

14. S. Cai, Y. Li, T. Li, and R. H. Deng. Attacks and improvements to an rifd mutual authentication protocol and its extensions. In *WISEC*, pages 51–58, Zurich, Switzerland, 2009.
15. D. Dolev and A. C. Yao. On the security of public key protocols. Technical report, Stanford, CA, USA, 1981.
16. J. Gattorna. Living Supply Chains. Pearson Education, 2006.
17. P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In *CT–RSA*, pages 163–178, San Francisco, California, USA, 2004.
18. A. Juels. Minimalist cryptography for low-cost RFID tags. In *SCN*, pages 149–164, Amalfi, Italyg, 2004.
19. A. Juels, R. Pappu, and B. Parno. Unidirectional Key Distribution Across Time and Space with Applications to RFID Security. In *USENIX*, pages 75–90, San Jose, California, USA, 2008.
20. H. Krawczyk, M. Bellare, and R. Canetti. RFC2104 - HMAC:Keyed-Hashing for Message Authentication. RFC Editor, 1997.
21. M. Langheinrich and R. Marti. Practical Minimalist Cryptography for RFID Privacy. *IEEE Systems Journal, Special Issue on RFID Technology*, 1(2):115–128, 2007.
22. M. Langheinrich and R. Marti. RFID privacy using spatially distributed shared secrets. In *UCS*, pages 1–16, Berlin, Heidelberg, 2007.
23. Y. Li and X. Ding. Protecting rfid communications in supply chains. In *ASIACCS*, pages 234–241, New York, NY, USA, 2007.
24. D. Molnar and D. Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *CCS*, pages 210–219, Washington, DC, USA, 2004.
25. G. Noubir, K. Vijayan, and H. J. Nussbaumer. Signature-based method for runtime fault detection in communication protocols. *Computer Communications*, 21:21–5, 1998.
26. I. S. Reed and G. Solomon. Polynomial codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.
27. B. Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption, Cambridge Security Workshop*, pages 191–204, London, UK, 1994.
28. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
29. B. Song and C. J. Mitchell. RFID Authentication Protocol for Low-cost Tags. In *WISEC*, pages 140–147, Alexandria, Virginia, USA, 2008.
30. D. Zanetti, L. Fellmann, S. Capkun. Privacy-preserving Clone Detection for RFID-enabled Supply Chains. In *IEEE RFID*, pages 37–44, Orlando, Florida, USA, 2010.