Secure and Private RFID-Enabled Third-Party Supply Chain Systems

Saiyu Qi, Yuanqing Zheng, *Member, IEEE*, Mo Li, Li Lu, *Member, IEEE*, and Yunhao Liu, *Fellow, IEEE*

Abstract—Radio Frequency Identification (RFID) is a key emerging technology for supply chain systems. By attaching RFID tags to various products, product-related data can be efficiently indexed, retrieved and shared among multiple participants involved in an RFID-enabled supply chain. The flexible data access property, however, raises security and privacy concerns. In this paper, we target at security and privacy issues in RFID-enabled supply chain systems. We investigate RFID-enabled Third-party Supply chain (RTS) systems and identify several inherent security and efficiency requirements. We further design a Secure RTS system called SRTS, which leverages RFID tags to deliver computation-lightweight crypto-IDs in the RTS system to meet both the security and efficiency requirements. SRTS introduces a Private Verifiable Signature (PVS) scheme to generate computation-lightweight crypto-IDs for product batches, and couples the primitive in RTS system through careful design. We conduct theoretical analysis and experiments to demonstrate the security and efficiency of SRTS.

Index Terms—Privacy, RFID, supply chain, production message

1 Introduction

Radio Frequency Identification (RFID) is a key emerging technology for supply chain systems. Compared with printed tags (e.g., barcodes, QR codes), RFID tags have moderate storage capacity to store unique IDs and support long-distance communication. By attaching tags to products, supply chain participants can read a tag to efficiently track the labeled product. The tag ID serves as an index to retrieve the product-related data from a database. Such an RFID-based supply chain facilitates information sharing among participants, enabling substantially improved product handling efficiency [1].

For instance, Toll Global Logistics, one of Asia's largest logistics providers, has adopted the RFID technology to track the tagged products of its served firms and cut labor costs [3]. The RFID infrastructure can be further leveraged to share product information with the participants involved in the supply chain. The sender stores IDs into tags and uploads the production messages indexed by the IDs in its database. The sender then delegates the logistics provider to deliver the tagged products to the receiver in a way that the latter two participants can flexibly read the tag IDs to retrieve the production messages of the labeled products from the sender's database.

- *S. Qi is with the School of Cyber Engineering, Xidian University, China. E-mail: syqi@connect.ust.hk.*
- *Y. Zheng is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. E-mail: csyqzheng@comp.polyu.edu.hk.*
- *M. Li is with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: limo@ntu.edu.sg.*
- *L. Lu is with the School of Computer Science and Engineering, University of Electronic Science and Technology, China. E-mail: luli2009@uestc.edu.cn.*
- *Y. Liu is with the TNLIST, School of Software, Tsinghua University, China. E-mail: yunhao@greenorbs.com.*

Manuscript received 6 July 2015; revised 27 Jan. 2016; accepted 8 Feb. 2016. Date of publication 3 Mar. 2016; date of current version 14 Oct. 2016. Recommended for acceptance by P.R. Schaumont.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2016.2538260

Despite the flexibility of data sharing enabled by RFID technology, it raises security and privacy concerns [4]. When tagged products flow in RTS system, the production messages stored in the sender's database should not be freely exchanged by the logistics provider and the receiver without any security guarantees. Given that the three participants typically belong to different trust sectors, different participants may have diverse requirements as shown in Fig. 1.

First, the sender and the receiver may be concerned about the privacy for the production messages of product batches against the logistics provider as the messages may be sensitive. Without privacy guarantee, a honest-but-curious logistics provider can collect production messages and explore non-trivial business secrets (e.g., production details, strategic relationships, buying interests of the receiver, etc). For instance, the logistics provider may use the collected messages together with out-of-bound information (e.g., product trading volume, transfer time, etc) to gradually infer the business transactions between the sender and the receiver. The sender and the receiver thus may be concerned about the privacy for the production messages of each delivered product batch, which we term as *batch privacy*.

Second, the logistics provider and the receiver may be concerned about non-repudiation for the production messages of product batches to prevent the sender from denying the creation of them. Without non-repudiation guarantee, a malicious sender may deny the creation of production messages to avoid economic loss. For instance, when a problematic product mismatched with its production message is found and needs to be recalled, a malicious sender may blame the logistics provider or the receiver, and refuse to recall the product. In fact, product delivery service is not always reliable in real trading systems. As exposed by China e-commerce complaints and rights of public service platform [5], customers receive inferior or fake products frequently in E-commerce business. The logistics provider and the receiver thus may be concerned about the non-repudiation for the production messages of each delivered product batch,

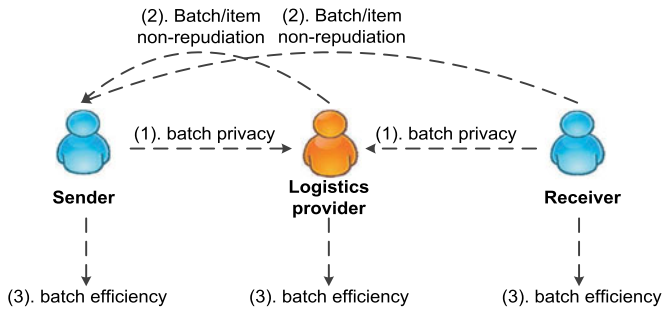


Fig. 1. Diverse requirements of different participants in an RFID-enabled RTS system.

so that they can prove the receipt of the whole batch or a certain product in the batch, which we term as *batch non-repudiation* and *item non-repudiation*, respectively.

Third, a large scale RTS system involves delivery of large amount of product batches. Ensuring privacy and non-repudiation for production messages of product batches among the three participants through crypto-tools may incur prohibitive communication and computation overhead. Specifically, when a product batch is delivered in the RTS system, the logistics provider may need to receive and process secured production messages for each product in the batch. As logistics provider is on the critical path of each product batch delivery, it easily becomes a bottleneck of the RTS system. The three participants thus are concerned about the delivery efficiency of product batches, which we term as *batch efficiency*.

In this paper, we target at security and efficiency issues in RFID-enabled Third-party Supply chain (RTS) system. We design SRTS, a Secure RTS system, to ensure the above three concerned requirements. Instead to directly exchange secured production messages through communication link, SRTS leverages RFID tags to deliver computation-lightweight crypto-IDs in the RTS system. Crypto-IDs serve as product IDs for product identification purpose as used in general RFID framework with the following two additional security properties: (1) crypto-IDs have non-repudiation property from which both the logistics provider and the receiver can acquire evidences to prove batch and item non-repudiations; and (2) crypto-IDs have privacy property to hide the content of the production messages. SRTS leverages the security properties of crypto-IDs as well as careful protocol design to achieve batch privacy, batch non-repudiation and item non-repudiation. By leveraging tags to distribute computation-lightweight crypto-IDs, SRTS also reduces the communication and computation overhead, achieving batch efficiency.

SRTS implements this idea through two steps. SRTS first introduces a Private Verifiable Signature (PVS) scheme to efficiently sign production messages as a whole in a privacy-preserving way. The signing result can be properly encoded into computation-lightweight crypto-IDs. SRTS then provides a set of distributed protocols to combine PVS scheme with RTS system through careful design. Specifically, product batch transfer protocol is executed in the delivery of a product batch. The sender generates crypto-IDs from the production messages of the batch through PVS scheme. After the delivery, both the logistics provider and the receiver acquire evidences from tag carried crypto-IDs. Later, the two parties can use the acquired evidences in a product batch arbitration protocol and an auditable item-level arbitration protocol to prove batch non-repudiation and item non-repudiation, respectively. The evidences are used in different ways in the two protocols to optimize the performance.

Our contributions can be summarized as follows. To our best knowledge, we are the first to propose efficient solutions to achieve these key security and efficiency requirements for large-scale RTS systems. We formulate and study three major security and efficiency requirements in RTS systems, i.e., batch privacy, batch non-repudiation/item non-repudiation and batch efficiency. We devise the SRTS scheme to achieve the desired requirements. We carry out extensive evaluation and evaluate the applicability of our approach on commodity C1G2 RFID systems.

2 Background and Problem

2.1 RFID Framework

Current RFID systems generally consist of three main components: RFID tags, RFID readers and a database. Lightweight commodity RFID tags harvest energy from RFID readers and backscatter incident signals to communicate with the RFID readers [1], [2]. The RFID tags have moderate storage capability with small onboard non-volatile memory, e.g., the Alien ALN-9640 passive RFID tags are equipped with a 512-bit user memory [28]. RFID readers can read/write a small amount of data (e.g., 512 bits) from/to user memory of RFID tags [28], [29]. Restricted by the small memory of RFID tags, product details are not carried by the tags but stored in the database, and accessed by the tag IDs as indexes to achieve fine-grained information sharing among participants. By labeling the products with RFID tags, supply chain participants can read tag IDs to efficiently track the labeled products and product details, which greatly facilitates the logistics and product management.

2.2 RFID-Enabled RTS System

An RFID-enabled RTS system consists of three participants: a logistics provider and two cooperative firms. We name the logistics provider as $\mathrm{Relaynode}$ and distinguish the two firms as Sender and $\mathrm{Receiver}$. Sender transfers tagged product batches to $\mathrm{Receiver}$ through $\mathrm{Relaynode}$. We describe detailed operations shortly.

To transfer a product batch, Sender attaches RFID tags to the batch. Depending on the applications, tags can be attached in different levels, such as item-level, packet-level or container-level. In this paper, we focus on item-level tag attachment as other levels can be easily extended from item-level.

Each product of the batch corresponds to a message $id_i||m_i$, which consists of an ID id_i and a production message m_i. Sender writes id_i into the attached tag and stores the message m_i in its database indexed by id_i. We consider a general case where the production messages of the products within the same batch may be different. For instance, a hospital ($\mathrm{Receiver}$) may order a batch of medicines from a medicine company (Sender), with the batch containing different types of medicines.

2.3 Desired Requirements

When a product batch is transferred through RTS system, different participants are concerned about different security

requirements against other participants for the batch. Yet, the three participants hope to afford lightweight batch delivery overhead. In summary, the participants are concerned about following three requirements:

- *Batch privacy:* Sender wants to share the production message m_i of each product in the batch with Receiver. As the content of m_i might be related to sensitive business matters, both Sender and Receiver do not want to leak m_i to Relaynode.
- *Batch and item non-repudiation:* Both Relaynode and Receiver want to obtain the ability to publicly prove batch non-repudiation—convincing an authority the receipt of the product batch with each product associated with a production message m_i; and item non-repudiation—convincing an authority the receipt of a certain product in the batch associated with a production message m_i.
- *Batch efficiency:* To achieve the above two security requirements, the production message m_i of each product in the batch needs to be properly equipped with privacy and non-repudiation properties and delivered from Sender to Relaynode and Receiver. As Relaynode is on the critical path of each product batch delivery, all the three participants hope to minimize the delivery overhead to avoid Relaynode becomes a bottleneck of the RTS system.

3 Overview of SRTS

Basically, SRTS combines cryptographic tools with RFID framework to achieve the desired security and efficiency requirements. SRTS leverages RFID tags to deliver computation-lightweight crypto-IDs in the RTS system to reduce the communication and computation overhead. For a product batch, each product tag is loaded with a crypto-ID and the corresponding production message is stored at Sender's database indexed by the crypto-ID. The crypto-ID serves as a product ID to identify the product as used in RFID framework with two additional security properties: (1) crypto-IDs have non-repudiation property from which both Relaynode and Receiver can acquire receipt evidences of a product batch. The constructed evidences can be used to prove batch and item non-repudiations for the product batch; and (2) the crypto-IDs have privacy property to hide the content of the production messages. As a result, the only way to acquire the production messages is to access Sender's database, which is only allowed by Receiver. In the following, we first give a strawman item-level solution, which presents partial design principles of SRTS. This solution then leads to our final design of SRTS.

3.1 A Strawman Item-Level Solution

A strawman design might be to encode a signed commitment as a cypto-ID, with the production message committed in the commitment. During the transfer of a product batch, both Relaynode and Receiver can directly collect the signed commitments from the product tags as evidences. Receiver is further allowed to use the signed commitments as indexes to retrieve the production messages from Sender's database. In an arbitration, Relaynode/Receiver could choose to reveal one or all of the collected signed commitments to an authority to prove item non-repudiation or batch non-repudiation, respectively. After convincing that the revealed signed commitment(s) is(are) correctly signed by Sender, the authority then requires Sender to reveal the committed production messages. Due to the security of commitment scheme, revealing tampered production messages will be detected by the authority.

This design, however, incurs prohibitive signature processing overhead. In a large-scale RTS system, where a large amount of product batches are delivered, the participants have to process tag carried digital signatures (contained in the signed commitment) in item-level. For a batch of n products, Sender needs to generate n signatures, and Relaynode and Receiver need to verify n signatures. Signature computation could incur considerable overhead (see detailed experiments in Section 6), and thus delay the transportation in RTS system. Besides, low-cost, storage-constrained tags cannot accommodate excessively long signatures, e.g., a 320-bit ECDSA signature alone may consume more than half of the commodity tag memory.

3.2 Our Design

Instead, our design of SRTS provides a new signature scheme called Private Verifiable Signature scheme, whose signing result can be encoded into computation-lightweight crypto-IDs. SRTS then provides a set of distributed protocols to combine PVS scheme with RFID framework through careful design.

PVS scheme. PVS scheme adopts a commit-then-sign pattern to sign the production messages of a product batch as a whole in two ways: public signing and private signing. In public signing, the production messages are committed into commitments and the concatenation of which are further signed by a digital signature scheme. The production messages as well as the signature is then output as a message batch-signature (MB) pair. In private signing, the production messages are committed and then signed in the same way as in the public signing, while the commitments as well as the signature is output as a commitment batch-signature (CB) pair. Both the MB and CB pairs incur constant signature storage and computation overhead regardless of the number of the production messages. PVS scheme guarantees that both the MB pair and the CB pair are unforgeable while the CB pair hides but binds the committed production messages. All these security properties are formally proved in security models.

Combining PVS scheme with RTS system. SRTS combines PVS scheme with RTS system by providing a set of distributed protocols. SRTS provides a product batch transfer protocol as shown in Fig. 2. To transfer a product batch, Sender privately signs the production messages of the batch to generate a CB pair and divides the pair into crypto-IDs. During the transfer, both Relaynode and Receiver can directly recover the CB pair from the tags as evidence. Additionally, Receiver can choose to retrieve the production messages from Sender, incorporate them with the CB pair to generate a MB pair, and stores the MB pair as its evidence.

SRTS provides a product batch arbitration protocol to prove batch non-repudiation. In the protocol, Relaynode/Receiver directly reveals its CB/MB pair of the batch to an authority to prove batch non-repudiation for the

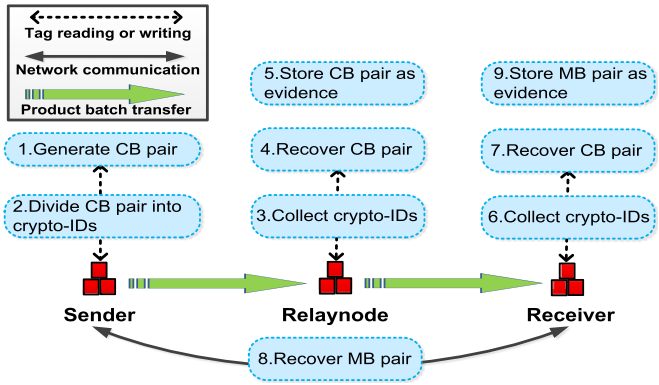


Fig. 2. Product batch transfer.

product batch. Due to the non-repudiation property of CB pair, the authority convinces that the commitments contained in the CB pair is generated from Sender. The authority further requires Sender to reveal these committed production messages. Due to the binding property of CB pair, revealing tampered production messages will be detected by the authority. Due to the non-repudiation property of MB pair, the authority directly convinces that the production messages contained in the MB pair is generated from Sender.

To efficiently prove item non-repudiation, SRTS provides an auditable item-level arbitration protocol. The protocol starts with a lightweight item non-repudiation proof phase which involves the exchange of attestations, and followed by an expensive audit phase which involves the reveal of a CB/MB pair. The protocol guarantees that if all participants behave correctly in the item non-repudiation proof phase, then the audit phase can be ignored. Whereas if misbehavior occurs, audit phase will be triggered and the malicious participant will be detected. Such a separation enforces all the participants behave in the lightweight item non-repudiation proof phase, and the expensive audit phase thus can be ignored.

4 SRTS: Private Verifiable Signature

In this section, we focus on PVS scheme and analyze its security properties. PVS scheme is the key component of SRTS to achieve lightweight batch-level signature processing (computation and storage) overhead.

With the PVS scheme, Sender can sign production messages to generate either an MB pair or a CB pair. The MB pair reveals the production messages, while the CB pair hides the production messages. Both the CB and MB pairs involve one signature regardless of the number of the production messages.

PVS scheme guarantees several security properties including: (1) both the MB pair and the CB pair are unforgeable, (2) the CB pair hides the production messages, and (3) the CB pair can only be opened to the hidden production messages.

4.1 Notations and Preliminaries

We list the important nations used in PVS scheme in Table 1. To sign product messages by PVS scheme, all the production messages need to be first concatenated to form a long message. At a high level, our construction adopts a multi-commit then single-sign mechanism to compose two crypto primitives: digital signature scheme and string commitment scheme.

TABLE 1
Important Notations

Notations	Definitions
(pk, sk)	Public-secret key pair of digital signature scheme
s	Signature of digital signature scheme
ck	Commitment key of string commitment scheme
com	Commitment of string commitment scheme
r	Random number used to generate commitment com
U	Domain of random number r
$\{str_i\}_{num}$	Concatenated batch of num messages $str_1 \|\| \cdots \|\| str_{num}$
(PK, SK)	Public-secret key pair of PVS scheme
$(\{m_i\}_n, \delta_{MB})$	Message batch-signature(MB) pair of PVS scheme
$\{m_i\}_n$	Concatenated batch of n production messages $m_1 \|\| \cdots \|\| m_n$
δ_{MB}	Signature in MB pair
$(\{mc_i\}_n, \delta_{CB})$	Commitment batch-signature(CB) pair of PVS scheme
$\{mc_i\}_n$	Concatenated batch of n PVS commitments $mc_1 \|\| \cdots \|\| mc_n$
δ_{CB}	Signature in CB pair

Digital signature. A digital signature scheme $\prod_D$ = (skg, sig, ver) consists of a key generation algorithm skg(), a signing algorithm sig() and a verification algorithm ver(). We require the signing algorithm to support the signing of variable length message. The security property of a signature scheme is that an adversary cannot forge a valid message-signature pair.

String commitment. A string commitment scheme $\prod_S$ = (ckg, commit) consists of a key generation algorithm ckg() and a committing algorithm commit(). A user can use commit() to generate a commitment for a string. The security properties of a commitment scheme are (1) hiding: the commitment does not leak any information about the string, and (2) binding: it is hard for the user to produce two different strings and a commitment such that the commitment is valid to both the strings.

4.2 Definition of PVS Scheme

By using the above two crypto primitives as building blocks, PVS scheme is constructed to provide six algorithms $\prod$ = (KeyGen, Sign, Verify, PriSign, PriVerify, Check). We briefly introduce the six algorithms as follows:

- KeyGen(λ) $\rightarrow$ (PK, SK): On input a security parameter λ, this algorithm outputs a public-secret key pair (PK, SK).
- Sign(PK, SK, $\{m_i\}_n$) $\rightarrow$ ($\{m_i\}_n$, δ_{MB}): On input a public key PK, a secret key SK and production messages $\{m_i\}_n$, this algorithm outputs an MB pair ($\{m_i\}_n$, δ_{MB}), which can be verified by Verify().
- Verify(PK, $\{m_i\}_n$, δ_{MB}) $\rightarrow$ (Accept, Reject): On input a public key PK and an MB pair ($\{m_i\}_n$, δ_{MB}), this

algorithm verifies the validity of ($\{m_i\}_n$, δ_{MB}) and outputs either Accept or Reject.

- PriSign(PK, SK, $\{m_i\}_n$) $\rightarrow$ ($\{mc_i\}_n$, δ_{CB}, $\{r_i\}_n$): On input a public key PK, a secret key SK and production messages $\{m_i\}_n$, this algorithm outputs a CB pair ($\{mc_i\}_n$, δ_{CB}) and witnesses $\{r_i\}_n$. The CB pair can be verified by PriVerify().
- PriVerify(PK, $\{mc_i\}_n$, δ_{CB}) $\rightarrow$ (Accept, Reject): On input a public key PK and a CB pair ($\{mc_i\}_n$, δ_{CB}), this algorithm verifies the validity of ($\{mc_i\}_n$, δ_{CB}) and outputs either Accept or Reject.
- Check(PK, $\{m_i\}_n$, $\{r_i\}_n$, $\{mc_i\}_n$, δ_{CB}) $\rightarrow$ (Accept, Reject): On input a public key PK, production messages $\{m_i\}_n$, witnesses $\{r_i\}_n$ and a CB pair ($\{mc_i\}_n$, δ_{CB}), this algorithm checks whether $\{m_i\}_n$ is the original production messages committed in ($\{mc_i\}_n$, δ_{CB}) and outputs either Accept or Reject.

4.3 Security Properties

The security of a PVS scheme is defined by four security properties: *MB-unforgeability*, *CB-unforgeability*, *Binding* and *Privacy*. Comparing with a general digital signature scheme, which only provides *MB-unforgeability*, a PVS scheme provides three additional security properties.

MB-unforgeability. Intuitively, *MB-unforgeability* means that it is computationally infeasible for an adversary $\mathcal{A}$ to forge a valid MB pair ($\{m_i\}_n$, δ_{MB}) with respect to the signer.

Definition 1 (MB-unforgeability). *A PVS scheme* $\prod$ = (KeyGen, Sign, Verify, PriSign, PriVerify, Check) *satisfies the* MB-unforgeability *property if every probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ has negligible advantage to win in the following experiment.*

Experiment $\mathbf{Exp}_{\mathcal{A}}^{MB-unf}$[PVS]:
(PK, SK) $\leftarrow$ KeyGen(λ);
($\{m_i\}_n^*$, $\delta^*{}_{MB}$) $\leftarrow \mathcal{A}^{\mathsf{Sign}(SK,PK,\perp)}(\lambda, PK)$;
output 1 if Verify(PK, $\{m_i\}_n^*$, δ_{MB}^*) $\rightarrow$ Accept
$\wedge \{m_i\}_n^* \notin M$;
else output 0

Our experiment allows $\mathcal{A}$ to submit message batches to a signing oracle Sign(SK, PK, $\perp$), which returns the corresponding MB pairs. All the queried message batches are recorded in a set M. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}_{\mathcal{A}}^{MB-unf}$[PVS] = Pr[$\mathbf{Exp}_{\mathcal{A}}^{MB-unf}$[PVS] $\Rightarrow$ 1].

CB-unforgeability. Intuitively, *CB-unforgeability* means that it is computationally infeasible for an adversary $\mathcal{A}$ to forge a valid CB pair ($\{mc_i\}_n$, δ_{CB}) with respect to the signer.

Definition 2 (CB-unforgeability). *A PVS scheme* $\prod$ = (KeyGen, Sign, Verify, PriSign, PriVerify, Check) *satisfies the* CB-unforgeability *property if every probabilistic polynomial time adversary $\mathcal{A}$ has negligible advantage to win in the following experiment.*

Experiment $\mathbf{Exp}_{\mathcal{A}}^{CB-unf}$[PVS]:
(PK, SK) $\leftarrow$ KeyGen(λ);
($\{mc_i\}_n^*$, $\delta^*{}_{CB}$) $\leftarrow \mathcal{A}^{\mathsf{PriSign}(SK,PK,\perp)}(\lambda, PK)$;
output 1 if PriVerify(PK, $\{mc_i\}_n^*$, δ_{CB}^*) $\rightarrow$ Accept
$\wedge \{mc_i\}_n^* \notin M$;
else output 0

Our experiment allows $\mathcal{A}$ to submit (message batch, witnesses) pairs to a signing oracle PriSign(SK, PK, $\perp$), which returns the corresponding CB pairs. The commitment batches contained in all the returned CB pairs are recorded in a set M. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}_{\mathcal{A}}^{CB-unf}$[PVS] = Pr[$\mathbf{Exp}_{\mathcal{A}}^{CB-unf}$[PVS] $\Rightarrow$ 1].

Binding. Intuitively, *binding* means that it is computationally infeasible for an adversary $\mathcal{A}$ to produce message batches $\{m_i\}_n \neq \{m_i'\}_n$ and a CB pair ($\{mc_i\}_n$, δ_{CB}), such that $\{mc_i\}_n$ is valid to both $\{m_i\}_n$ and $\{m_i'\}_n$.

Definition 3. (binding) *A PVS scheme* $\prod$ = (KeyGen, Sign, Verify, PriSign, PriVerify, Check) *satisfies the* binding *property if every probabilistic polynomial time adversary $\mathcal{A}$ has negligible advantage to win in the following experiment.*

Experiment $\mathbf{Exp}_{\mathcal{A}}^{bind}$[PVS]:
(PK, SK) $\leftarrow$ KeyGen(λ);
($\{m_i\}^1{}_n$, $\{r_i\}^1{}_n$, $\{m_i\}^2{}_n$, $\{r_i\}^2{}_n$, $\{mc_i\}_n$, δ_{CB}) $\leftarrow$
$\leftarrow \mathcal{A}(\lambda, PK, SK)$;
output 1 if Check(PK, $\{m_i\}^1{}_n$, $\{r_i\}^1{}_n$, $\{mc_i\}_n$, δ_{CB})
$\rightarrow$ Accept
$\wedge$ Check(PK, $\{m_i\}^2{}_n$, $\{r_i\}^2{}_n$, $\{mc_i\}_n$, δ_{CB})
$\rightarrow$ Accept
$\wedge \{m_i\}^1{}_n \neq \{m_i\}^2{}_n$;
else output 0

We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}_{\mathcal{A}}^{bind}$[PVS] = Pr [$\mathbf{Exp}_{\mathcal{A}}^{bind}$[PVS] $\Rightarrow$ 1].

Privacy. Intuitively, *privacy* means that it is computationally infeasible for an adversary $\mathcal{A}$ to learn non-trivial knowledge about the message batch $\{m_i\}_n$ from a CB pair ($\{mc_i\}_n$, δ_{CB}).

Definition 4 (privacy). *A PVS scheme* $\prod$ = (KeyGen, Sign, Verify, PriSign, PriVerify, Check) *satisfies the* privacy *property if every probabilistic polynomial time adversary $\mathcal{A}$ has negligible advantage to win in the following experiment.*

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Priv}$[PVS]:
(PK, SK) $\leftarrow$ KeyGen(λ);
($\{m_i\}_n^1$, $\{m_i\}_n^2$) $\leftarrow \mathcal{A}(\lambda, PK)$;
$b \xleftarrow{R} \{0, 1\}$;
($\{mc_i\}_n^b$, δ_{CB}, $\{r_i\}_n^b$) $\leftarrow$ PriSign(PK, SK, $\{m_i\}_n^b$);
$b' \leftarrow \mathcal{A}(\lambda, PK, \{mc_i\}_n^b, \delta_{CB})$;
output 1 if b'=b, else output 0

Our experiment uses indistinguishability of multiple messages, which allows $\mathcal{A}$ to submit two message batches and get a CB pair. The committed messages are chosen randomly from one of the two message batches. $\mathcal{A}$ then guesses which message batch is committed. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}_{\mathcal{A}}^{Priv}$[PVS] = Pr[$\mathbf{Exp}_{\mathcal{A}}^{Priv}$[PVS] $\Rightarrow$ 1] - 1/2.

4.4 Construction of PVS Scheme

We describe a concrete PVS scheme in Table 2. Next, we analyze the security of our construction.

Theorem 1. *If the digital signature scheme* $\prod_D$ *is* unforgeable *and if the string commitment scheme* $\prod_S$ *is* binding, *then our construction of PVS scheme* $\prod$ *achieves* MB-unforgeability.

Proof. When $\mathcal{A}$ submits a message batch $\{m_i\}_n$ to the signing oracle Sign(SK, PK, $\perp$), the oracle computes a commitment

TABLE 2
Construction of Private Verifiable Signature

KeyGen(λ) $\rightarrow$ (PK, SK):
— (pk, sk) $\leftarrow$ sk$g$(λ);
— $ck \leftarrow$ ck$g$(λ);
— Output PK = (pk, ck) and SK = sk.

Sign(PK, SK, $\{m_i\}_n$) $\rightarrow$ ($\{m_i\}_n$, δ_{MB}):
— for $1 \leq i \leq n$:
— $r_i \xleftarrow{R} U$;
— $com_i \leftarrow$ commit(ck, m_i, r_i);
— $s \leftarrow$ sig(sk, $\{com_i\}_n$);
— δ_{MB} = (s, $\{r_i\}_n$);
— Output a MB pair ($\{m_i\}_n$, δ_{MB}).

Verify(PK, $\{m_i\}_n$, δ_{MB}) $\rightarrow$ (Accept, Reject):
— for $1 \leq i \leq n$:
— $com_i \leftarrow$ commit(ck, m_i, r_i);
— ver(pk, s, $\{com_i\}_n$) = valid/invalid?
— If valid, output Accept, else output Reject.

PriSign(PK, SK, $\{m_i\}_n$) $\rightarrow$ ($\{mc_i\}_n$, δ_{CB}, $\{r_i\}_n$):
— for $1 \leq i \leq n$:
— $r_i \xleftarrow{R} U$;
— $com_i \leftarrow$ commit(ck, m_i, r_i);
— $s \leftarrow$ sig(sk, $\{com_i\}_n$);
— $\{mc_i\}_n = \{com_i\}_n$;
— $\delta_{CB} = s$;
— Output a CB pair ($\{mc_i\}_n$, δ_{CB}) and witnesses $\{r_i\}_n$.

PriVerify(PK, $\{mc_i\}_n$, δ_{CB}) $\rightarrow$ (Accept, Reject):
— ver(pk, δ_{CB}, $\{mc_i\}_n$) = valid/invalid?
— If valid, output Accept, else output Reject.

Check(PK, $\{m_i\}_n$, $\{r_i\}_n$, $\{mc_i\}_n$, δ_{CB}) $\rightarrow$ (Accept, Reject):
— for $1 \leq i \leq n$:
— mc_i = commit(ck, m_i, r_i)?
— If all equal, output Accept, else output Reject.

batch $\{mc_i\}_n$ for $\{m_i\}_n$, signs $\{mc_i\}_n$ using the underlying digital signature scheme, and returns an MB-pair ($\{m_i\}_n$, δ_{MB}). Suppose $\mathcal{A}$ queried l message batches ($\{m_i\}^1_{n^1}$, ..., $\{m_i\}^l_{n^l}$) to the signing oracle Sign(SK, PK, $\perp$), and ($\{m_i\}^*_n$, $\delta^*{}_{MB}$) is the purported forgery output by $\mathcal{A}$. The forgery must fall in at least one of the following two cases: (1) Case 1: For every message batch $\{m_i\}^j_{n^j} (1 \leq j \leq l)$ submitted by $\mathcal{A}$, the corresponding commitment batch $\{mc_i\}^j_{n^j}$ is different from the commitment batch $\{mc_i\}^*_n$ of $\{m_i\}^*_n$. (2) Case 2: There is a message batch $\{m_i\}^j_{n^j}$ ($1 \leq j \leq l$) submitted by $\mathcal{A}$ such that its commitment batch $\{mc_i\}^j_{n^j}$ equals $\{mc_i\}^*_n$. We thus conclude $\mathbf{Adv}^{MB-unf}_{\mathcal{A}}$[PVS] $\leq$ Pr[Case 1] + Pr[Case 2].

In the first case, the adversary $\mathcal{A}$ could be used to build an adversary $\mathcal{B}$ to break *unforgeability* of the digital signature scheme. At the beginning, $\mathcal{B}$ is given a public key pk of the digital signature scheme and generates a commitment key ck of the string commitment scheme. $\mathcal{B}$ then generates a public key of PVS scheme PK = (pk, ck) and gives PK to $\mathcal{A}$. $\mathcal{B}$ simulates the oracle Sign(PK, SK, $\perp$) as follows. When $\mathcal{A}$ queries a message batch $\{m_i\}^j_{n^j} (1 \leq j \leq l)$, $\mathcal{B}$ generates witnesses $\{r_i\}^j_{n^j}$ and uses this batch as well as ck to generate a commitment batch $\{mc_i\}^j_{n^j}$ for $\{m_i\}^j_{n^j}$. $\mathcal{B}$ then submits $\{mc_i\}^j_{n^j}$ to its own signing oracle sig(sk, $\perp$) of the digital signature scheme to get a signature s^j. Finally, $\mathcal{B}$ returns an MB-pair ($\{m_i\}^j_{n^j}$, δ_{MB}) to $\mathcal{A}$, where δ_{MB} = (s^j, $\{r_i\}^j_{n^j}$). When $\mathcal{A}$ outputs ($\{m_i\}^*_n$, $\delta^*{}_{MB}$), $\mathcal{B}$ parses $\delta^*{}_{MB}$ = (s^*, $\{r_i\}^*_n$), computes a commitment batch $\{mc_i\}^*_n$ from $\{m_i\}^*_n$ and $\{r_i\}^*_n$, and outputs a message-signature pair ($\{mc_i\}^*_n$, s^*). Obviously, Pr[Case 1] equals the probability of $\mathcal{B}$ to break *unforgeability* of the digital signature, which happens with negligible probability.

In the second case, the adversary $\mathcal{A}$ could be used to build an adversary $\mathcal{B}$ to break *binding* of the string commitment scheme. At the beginning, $\mathcal{B}$ is given a commitment key ck of the string commitment scheme and generates a public-secret key pair (pk, sk) of the digital signature scheme. $\mathcal{B}$ then generates a public key of PVS scheme PK = (pk, ck) and gives PK to $\mathcal{A}$. $\mathcal{B}$ simulates the oracle Sign(PK, SK, $\perp$) as follows. When $\mathcal{A}$ queries a message batch $\{m_i\}^j_{n^j} (1 \leq j \leq l)$, $\mathcal{B}$ generates witnesses $\{r_i\}^j_{n^j}$ and uses $\{r_i\}^j_{n^j}$ as well as ck to generate a commitment batch $\{mc_i\}^j_{n^j}$ for $\{m_i\}^j_{n^j}$. $\mathcal{B}$ then signs $\{mc_i\}^j_{n^j}$ by using sk to get a signature s^j. Finally, $\mathcal{B}$ returns an MB-pair ($\{m_i\}^j_{n^j}$, δ_{MB}) to $\mathcal{A}$, where δ_{MB} = (s^j, $\{r_i\}^j_{n^j}$). When $\mathcal{A}$ outputs ($\{m_i\}^*_n$, δ^*_{MB}), $\mathcal{B}$ finds the message batch $\{m_i\}^j_{n^j} (1 \leq j \leq l)$ submitted by $\mathcal{A}$ with $\{m_i\}^j_{n^j} \neq \{m_i\}^*_n$ but $\{mc_i\}^j_{n^j} = \{mc_i\}^*_n$. $\mathcal{B}$ then finds a message $m_i \in \{m_i\}^j_{n^j}$ and a message $m'_i \in \{m_i\}^*_n$ satisfying the condition commit(ck, m_i, r_i) = commit(ck, m'_i, r'_i) and outputs (m_i, r_i, m'_i, r'_i). Obviously, Pr[Case 2] equals the probability of $\mathcal{B}$ to break *binding* of the string commitment scheme, which is negligible. □

Theorem 2. *If the digital signature scheme* $\prod_D$ *is* unforgeable, *then our construction of PVS scheme* $\prod$ *achieves* CB-unforgeability.

Proof. The adversary $\mathcal{A}$ could be used to construct an adversary $\mathcal{B}$ to break *unforgeability* of the digital signature scheme. At the beginning, $\mathcal{B}$ is given a public key pk of the digital signature scheme and generates a commitment key ck of the string commitment scheme. $\mathcal{B}$ then generates a public key of PVS scheme PK = (pk, ck) and gives PK to $\mathcal{A}$. $\mathcal{B}$ simulates the oracle PriSign(SK, PK, $\perp$) as follows. When $\mathcal{A}$ queries a message batch $\{m_i\}^j_{n^j}$ and witnesses $\{r_i\}^j_{n^j}$, $\mathcal{B}$ uses $\{r_i\}^j_{n^j}$ as well as ck to generate a commitment batch $\{mc_i\}^j_{n^j}$ for $\{m_i\}^j_{n^j}$. $\mathcal{B}$ further queries $\{mc_i\}^j_{n^j}$ to its own signing oracle sig(sk, $\perp$) to get a signature s^j. $\mathcal{B}$ then returns a CB-pair ($\{mc_i\}^j_{n^j}$, δ^j_{CB}) to $\mathcal{A}$, where $\delta^j_{CB} = s^j$. When $\mathcal{A}$ outputs ($\{mc_i\}^*_n$, $\delta^*{}_{CB}$), $\mathcal{B}$ directly outputs ($\{mc_i\}^*_n$, $\delta^*{}_{CB}$). Obviously, $\mathbf{Adv}^{CB-unf}_{\mathcal{A}}$[PVS] equals the probability of $\mathcal{B}$ to break *unforgeability* of the digital signature scheme, which is negligible. □

Theorem 3. *If the string commitment scheme* $\prod_S$ *is* binding, *then our construction of PVS scheme* $\prod$ *achieves* binding.

Proof. The adversary $\mathcal{A}$ could be used to construct an adversary $\mathcal{B}$ to break *binding* of the string commitment scheme. At the beginning, $\mathcal{B}$ is given a commitment key ck of the string commitment scheme and generates a public-secret key pair (pk, sk) of the digital signature scheme. $\mathcal{B}$ then

generates a public-secret key pair (PK, SK) of PVS scheme with $PK = (pk, ck)$ and $SK = sk$ and gives (PK, SK) to $\mathcal{A}$. Suppose $\mathcal{A}$ outputs $(\{m_i\}^1_n, \{r_i\}^1_n, \{m_i\}^2_n, \{r_i\}^2_n, \{mc_i\}_n, \delta_{CB})$ satisfying the condition $\mathsf{Check}(PK, \{m_i\}^1_n, \{r_i\}^1_n, \{mc_i\}_n, \delta_{CB}) \rightarrow \mathsf{Accept} \wedge \mathsf{Check}(PK, \{m_i\}^2_n, \{r_i\}^2_n, \{mc_i\}_n, \delta_{CB}) \rightarrow \mathsf{Accept} \wedge \{m_i\}^1_n \neq \{m_i\}^2_n$. $\mathcal{B}$ finds $m_i \in \{m_i\}^1_n$ and $m'_i \in \{m_i\}^2_n$ satisfying the condition $\mathsf{commit}(ck, m_i, r_i) = \mathsf{commit}(ck, m'_i, r'_i) \wedge m_i \neq m'_i$ and outputs (m_i, r_i, m'_i, r'_i). Obviously, $\mathbf{Adv}^{bind}_{\mathcal{A}}[\mathrm{PVS}]$ equals the probability of $\mathcal{B}$ to break *binding* of the string commitment scheme, which is negligible. □

Theorem 4. *If the string commitment scheme* $\prod_S$ *is* hiding, *then our construction of PVS scheme* $\prod$ *achieves* privacy.

Proof. The adversary $\mathcal{A}$ could be used to construct an adversary $\mathcal{B}$ to break *hiding* (semantic security of multiple messages) of the string commitment scheme. At the beginning, $\mathcal{B}$ is given a commitment key ck of the string commitment scheme and generates a public-secret key pair (pk, sk) of the digital signature scheme. $\mathcal{B}$ then generates a public key of PVS scheme $PK = (pk, ck)$ and gives PK to $\mathcal{A}$. $\mathcal{B}$ simulates the oracle $\mathsf{PriSign}(PK, SK, \perp)$ as follows. When $\mathcal{A}$ submits $(\{m_i\}^1_n, \{m_i\}^2_n)$, $\mathcal{B}$ submits $(\{m_i\}^1_n, \{m_i\}^2_n)$ to its own committing oracle of the string commitment scheme to get a commitment batch $\{mc_i\}^b_n$. $\mathcal{B}$ then runs $s \leftarrow \mathsf{sig}(\{mc_i\}^b_n, sk)$, sets $\delta_{CB} = s$, and returns $(\{mc_i\}^b_n, \delta_{CB})$ to $\mathcal{A}$. Suppose $\mathcal{A}$ outputs b'. $\mathcal{B}$ also outputs b' as its guess. Obviously, $\mathbf{Adv}^{priv}_{\mathcal{A}}[\mathrm{PVS}]$ equals the advantage of $\mathcal{B}$ to break *hiding* of the string commitment scheme, which is negligible. □

5 SRTS: Protocol Design

In this section, we focus on SRTS. It consists of three protocols: *product batch transfer*, *product batch arbitration* and *auditable item-level arbitration*.

5.1 Initialization

In SRTS, the three participants leverage existing secure network communication protocols (such as SSL/TLS) to achieve reliable message exchange. When two participants need to exchange messages, they first authenticate the identity of each other and then establish a secure channel to exchange messages.

Both the Relaynode and Receiver need to have their own public-private key pairs of digital signature scheme. Their public keys need to be certified by the key authority and published, so that anyone can verify the validity of their signatures.

To guarantee the security of SRTS, Sender needs to generate a public-secret key pair (PK, SK) of PVS scheme and requests a certificate for PK from a key authority so that anyone can verify the validity of PK. Recall that a PK consists of a public key pk of a digital signature scheme and a commitment key ck of a string commitment scheme. The commitment scheme requires ck to be correctly generated by a trustworthy party to guarantee its security properties. To achieve this, we require the key authority to only accept pk from Sender, generates ck by itself to form a public key $PK = (pk, ck)$, and issues a certificate on PK.

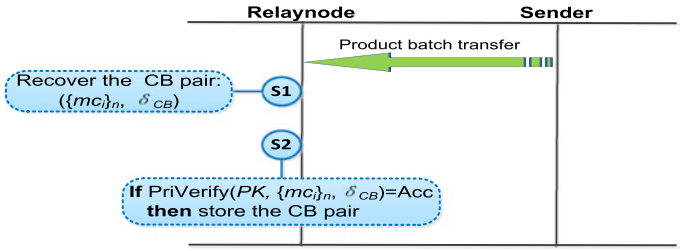


Fig. 3. Product batch transfer in Relaynode case.

5.2 Product Batch Transfer

5.2.1 Message Batch Processing

To transfer a product batch of n products, Sender generates the production messages $\{m_i\}_n$ for the product batch. Sender then generates n crypto-IDs for $\{m_i\}_n$ through steps S1-S3:

S1. Sender runs the $\mathsf{PriSign}()$ algorithm of PVS scheme to generate a CB pair for $\{m_i\}_n$ and witnesses:

$$(\{mc_i\}_n, \delta_{CB}, \{r_i\}_n) \leftarrow \mathsf{PriSign}(PK, SK, \{m_i\}_n)$$

S2. Consider the commitment batch $\{mc_i\}_n$ in the CB pair. Notice that $\{mc_i\}_n$ represents a long message $mc_1||mc_2||...||mc_n$. For each commitment mc_i, Sender concatenates mc_i with an in-batch index i, where i is the position of mc_i in the long message. Sender then encodes each indexed commitment $i||mc_i$ as a crypto-ID id_i and stores the n crypto-IDs into the n product tags. Sender attaches a batch tag to the product batch and stores the common signature δ_{CB} into it. Notice that a commitment mc_i is actually a random element of a group G, which can properly serve as a general ID to uniquely identify a tag.

S3. Sender creates a batch record in its database. The batch record contains a two-layer index structure. The common signature δ_{CB} of the CB pair is used as the first-layer index for the whole batch record. The crypto-IDs are used as the second-layer index for the individual elements of both the production messages and the witnesses:

$$(\delta_{CB}, \{id_i||m_i\}_n, \{id_i||r_i\}_n).$$

5.2.2 Sender → Relaynode

The process is shown in Fig. 3 and described as follows. Sender directly transfers the product batch to Relaynode. Upon receiving it, Relaynode can use the tag carried cypto-IDs to identify and track each product in the batch. Relaynode can also collect the cypto-IDs from the product tags as well as the common signature from the batch tag to recover a CB pair through steps S1-S2:

S1. Relaynode concatenates the collected cypto-IDs following the order of their concatenated in-batch indexes to recover a commitment batch $\{mc_i\}_n$:

$$\{mc_i\}_n = mc_1||mc_2||\ldots||mc_n.$$

Relaynode then combines the commitment batch with the collected common signature to recover the CB pair:

$$(\{mc_i\}_n, \delta_{CB}).$$

S2. Relaynode runs the $\mathsf{PriVerify}()$ algorithm of PVS scheme to verify if the CB pair is valid:

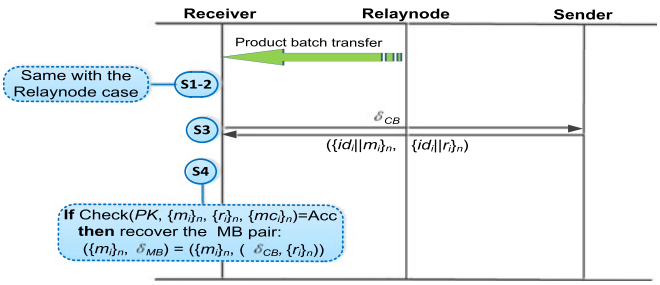


Fig. 4. Product batch transfer in Receiver case.

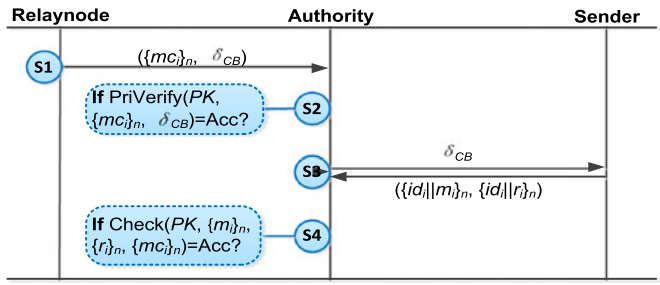


Fig. 5. Product batch arbitration in Relaynode case.

$$\text{PriVerify}(PK, \{mc_i\}_n, \delta_{CB}) = \text{Accept?}$$

If valid, Relaynode stores the CB pair $(\{mc_i\}_n, \delta_{CB})$ as an evidence in its database. Later, Relaynode transfers the product batch to Receiver.

5.2.3 Relaynode → Receiver

The process is shown in Fig. 4 and described as follows. Upon receiving the product batch from Relaynode, Receiver can use the tag carried cypto-IDs to identify and track each product in the batch. Receiver can also collect the cypto-IDs from the product tags as well as the common signature from the batch tag to recover an MB pair through steps S1-S4:

S1-S2. Similar with Relaynode, Receiver recovers a CB pair and runs the PriVerify() algorithm of PVS scheme to verify its validity.

S3. If the CB pair $(\{mc_i\}_n, \delta_{CB})$ is valid, Receiver returns the common signature δ_{CB} in the CB pair to Sender to fetch the production messages and the witnesses:

$$(\{id_i||m_i\}_n, \{id_i||r_i\}_n).$$

S4. Receiver runs the Check() algorithm of PVS scheme to verify if the fetched production messages are the exact ones committed in the CB pair:

$$\text{Check}(PK, \{m_i\}_n, \{r_i\}_n, \{mc_i\}_n) = \text{Accept?}$$

If valid, Receiver recovers an MB pair from the CB pair:

$$(\{m_i\}_n, \delta_{MB}) = (\{m_i\}_n, (\delta_{CB}, \{r_i\}_n))$$

and stores the MB pair as an evidence in its database.

After the above four steps, Receiver accepts the fetched production messages $\{id_i||m_i\}_n$ as valid for the received product batch. For each product with tag carried crypto-ID id_i, Receiver can easily search the corresponding production message m_i from $\{id_i||m_i\}_n$.

5.3 Product Batch Arbitration

5.3.1 Relaynode Case

The process is shown in Fig. 5 and described as follows. To prove batch non-repudiation for the product batch, Relaynode starts an arbitration with the authority through steps S1-S4:

S1. Relaynode sends its evidence $(\{mc_i\}_n, \delta_{CB})$ (the CB pair) of the product batch to the authority.

S2. The authority runs the PriVerify() algorithm of PVS scheme to verify the evidence:

$$\text{PriVerify}(PK, \{mc_i\}_n, \delta_{CB}) = \text{Accept?}$$

S3. If Accept, the authority returns the common signature δ_{CB} in the CB pair to Sender to fetch the production messages and the witnesses of the product batch:

$$(\{id_i||m_i\}_n, \{id_i||r_i\}_n).$$

S4. The authority runs the Check() algorithm of PVS scheme to verify if the fetched production messages are the exact ones committed in the evidence:

$$\text{Check}(PK, \{m_i\}_n, \{r_i\}_n, \{mc_i\}_n) = \text{Accept?}$$

If Accept, the authority convinces the receipt of a product batch with each product associated with a production message $m_i \in \{m_i\}_n$.

5.3.2 Receiver Case

To prove batch non-repudiation for the product batch, Receiver starts an arbitration with the authority through steps S1-S2:

S1. Receiver sends its evidence $(\{m_i\}_n, \delta_{MB})$ (the MB pair) of the product batch to the authority.

S2. The authority runs the algorithm Verify() of PVS scheme to verify the evidence:

$$\text{Verify}(PK, \{m_i\}_n, \delta_{MB}) \rightarrow \text{Accept?}$$

If Accept, the authority convinces the receipt of a product batch with each product associated with a production message $m_i \in \{m_i\}_n$.

5.4 Auditable Item-Level Arbitration

Auditable item-level arbitration supports a more flexible scenario where Relaynode/Receiver may want to prove item non-repudiation, i.e., receipt of a certain product (in a product batch) from Sender associated with a production message m_i to the authority. The protocol involves exchange of attestations, which are signed messages. For simplicity, we use the notion Sign(m) to denote a signed message m as well as the corresponding signature.

5.4.1 Relaynode Case

The process is shown in Fig. 6 and described as follows. To prove item non-repudiation for a certain product, Relaynode starts an arbitration with the authority through steps S1-S4 (S1-S3 belong to item non-repudiation proof phase and S4 belongs to audit phase):

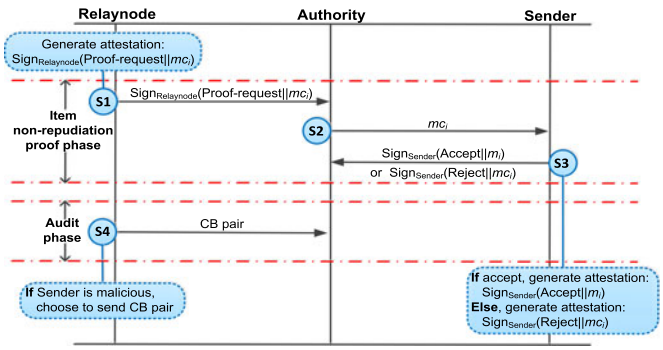


Fig. 6. Auditable item-level arbitration in Relaynode case.

S1. Relaynode concatenates a notion Proof-request with the commitment mc_i of the product, and signs the whole message to generate an attestation:

$$\mathrm{Sign}_{\mathrm{Relaynode}}(\textsf{Proof-request}||mc_i).$$

Relaynode then sends this attestation to the authority. Notice that mc_i is contained in the CB pair stored by Relaynode.

S2. Upon receiving the attestation, the authority records it, sends the commitment mc_i to Sender and requests the latter to reveal the production message committed in mc_i.

S3. Upon receiving mc_i, Sender decides whether to accept the authority's request. If accept, Sender concatenates a notion Accept with a production message m_i, and signs the whole message to generate an attestation:

$$\mathrm{Sign}_{\mathrm{Sender}}(\textsf{Accept}||m_i).$$

If reject, Sender concatenates a notion Reject with the commitment mc_i, and signs the whole message to generate an attestation:

$$\mathrm{Sign}_{\mathrm{Sender}}(\textsf{Reject}||mc_i).$$

Sender then returns its attestation to the authority.

On the other hand, the authority either checks if m_i is the exact production message committed in mc_i and records the attestation. If the first case happens, the authority convinces the receipt of a product associated with the production message m_i. Finally, the authority informs the proof result to Relaynode.

S4. If a malicious Sender refuses to reveal the production message for a valid commitment mc_i, then Relaynode can choose to send the CB pair, which includes mc_i, to the authority to prove the validity of mc_i.

Efficiency. The design of the audit phase S4 enforces both Sender and Relaynode to correctly execute the item non-repudiation proof phase S1-S3; otherwise their malicious behaviours will be detected by the authority through S4. But if Sender and Relaynode correctly execute item non-repudiation proof phase, *auditable item-level arbitration protocol* will end and S4 will not be executed. From Fig. 6, we can clearly see that the item non-repudiation proof phase only involves the commitment or the production message of the targeted product, and the CB pair of a product batch is not involved. Overall, we can conclude that our protocol in Relaynode case prunes the abundant overhead incurred by the rest products in the same batch.

5.4.2 Receiver Case

Receiver can prove item non-repudiation for a certain product through four steps similar with the Relaynode case.

S1. Receiver sends an attestation:

$$\mathrm{Sign}_{\mathrm{Receiver}}(\textsf{Proof-request}||m_i)$$

to the authority. Note that m_i is contained in the MB pair stored by Receiver.

S2. Upon receiving the attestation, the authority records it, sends the message m_i to Sender and requests the latter to decide if to accept the message.

S3. If Sender decides to accept, it directly returns an attestation:

$$\mathrm{Sign}_{\mathrm{Sender}}(\textsf{Accept}||m_i)$$

to the authority. Otherwise, Sender returns an attestation:

$$\mathrm{Sign}_{\mathrm{Sender}}(\textsf{Reject}||m_i)$$

to the authority.

S4. Receiver sends the MB pair, which includes m_i, to the authority to prove the validity of m_i.

Efficiency. The efficiency is similar with the Relaynode case, and we just ignore it here. Clearly, we can also conclude that our protocol in Receiver case prunes the abundant overhead incurred by the rest products (and protects the privacy of their production messages) in the same batch.

5.5 Security Analysis

We now analyze the security of SRTS. Specifically, we focus on *Batch privacy* and *Batch and Item non-repudiation*.

5.5.1 Batch Privacy

Consider the product batch transfer protocol of SRTS in which a product batch is transferred. Sender only stores a CB pair into the tags of the product batch. Due to *privacy* property of PVS scheme, it is infeasible for Relaynode to learn any non-trivial knowledge about the production messages from the CB pair. Formally, Relaynode can be described as the adversary defined in *Definition 4* (*privacy*) and the security is proved in *Theorem 4*.

5.5.2 Batch Non-Repudiation

We analyze *Batch non-repudiation* in Relaynode case and Receiver case separately as follows.

Relaynode case. Consider the Relaynode case of the product batch arbitration protocol in SRTS. Suppose Relaynode wants to prove batch non-repudiation for a product batch. Relaynode shows the CB pair of the product batch for the authority to verify. Due to *CB unforgeability* of PVS scheme, the authority convinces receipt of a product batch with each product associated with a product message committed in mc_i. Formally, Relaynode can be described as a weak version of the adversary defined in *Definition 2* (*CB-unforgeability*) and the security is proved in *Theorem 2*.

The authority then requests Sender to reveal the production messages $\{m_i\}_n$ committed in the CB pair. Due to *binding* property of PVS scheme, Sender cannot reveal tampered production messages $\{m'_i\}_n$ which can also pass the check.

Formally, Sender can be described as the adversary defined in *Definition 3* (*binding*) and the security is proved in *Theorem 3*.

Combining the above two steps, the authority convinces receipt of a product batch with each product associated with a production message $m_i \in \{m_i\}_n$.

Receiver case. Consider the Receiver case of the product batch arbitration protocol in SRTS. Suppose Receiver wants to prove batch non-repudiation for a product batch. Receiver shows the MB pair of the product batch for the authority to verify. Due to *MB unforgeability* of PVS scheme, the authority convinces receipt of a product batch with each product associated with a production message m_i contained in the MB pair. Formally, Receiver can be described as the adversary defined in *Definition 1* (*MB-unforgeability*) and the security is proved in *Theorem 1*.

5.5.3 Item Non-Repudiation

We analyze *Item non-repudiation* in Relaynode case and Receiver case separately as follows.

Relaynode case. Consider the Relaynode case of the auditable item-level arbitration protocol in SRTS. Suppose Relaynode wants to prove item non-repudiation for a certain product. The audit phase S4 enforces a malicious Sender to accept the revealing request for a valid commitment; since if the malicious Sender chooses to reject, then it must provide a reject attestation in S3 to the authority. In this case, Relaynode shows the CB pair to verify the commitment in S4. Due to *CB unforgeability* of PVS scheme, the authority confirms that the commitment is valid. As the reject attestation is signed by Sender, due to *unforgeability* of signature scheme, the authority can use this attestation to accuse that Sender rejects a valid commitment.

The audit phase S4 also enforces a malicious Relaynode to prove item non-repudiation for a valid product; since if Relaynode provides a fake commitment in S1, Sender can safely reject to reveal the production message committed in the commitment. In this case, the malicious Relaynode cannot show a valid CB pair to verify the fake commitment in S4 due to *CB unforgeability* of PVS scheme. Recall that Relaynode already provided a request attestation in S1 to the authority, which is signed by Relaynode. Due to *unforgeability* of signature scheme, this request attestation can be used by the authority to accuse that Relaynode requests to prove item non-repudiation for a fake product (commitment).

Receiver case. Consider the Receiver case of the auditable item-level arbitration protocol in SRTS. The analysis is similar with the Relaynode case, and we just ignore it here.

6 Performance Evaluation

6.1 Selection of Crypto-Primitives

Recall that PVS scheme builds on top of two crypto-primitives: namely digital signature scheme and string commitment scheme. We now consider several instantiations of the two crypto-primitives that can be used to implement PVS scheme.

BLS signature scheme. We consider BLS signature scheme [33] as an instantiation of the digital signature scheme. In this scheme, a secret key is a random value x selected from an interval [0, q-1] where q is a prime number. The corresponding public key is g^x where g is a generator of a group G with order q. To sign a message m, one computes $h=H(m)$ where $H()$ is a hash function hashing m to an element of G, and computes a signature $sig=h^x$. Given $pk=g^x$ and $sig=h^x$, the verification of the signature sig is performed by checking the equivalence $e(sig, g)=e(H(m), g^x)$. Here, $e(\cdot,\cdot)$ is a bilinear map $G\times G\rightarrow G_T$ mapping two elements of group G to an element of group G_T.

Pedersen commitment scheme. We consider Pedersen commitment scheme [26] as an instantiation of the string commitment scheme. In this scheme, a commitment key ck comprises a generator g of a group G with prime order q and a random element h of G. To commit to a string m in an interval [0, q-1], one draws a random value r in [0, q-1] and computes the commitment $com=g^m h^r$. To open a commitment, one sends the tuple (m, r) to the verifier who checks whether $com=g^m h^r$.

Hash commitment scheme. We consider Hash commitment scheme as another instantiation of the string commitment scheme, which can be used to replace Pedersen commitment scheme with security-efficiency tradeoff. A commitment of Hash commitment scheme is a hash value H(m, r), where H() is a collision-resistant hash function, m is the committed string and r is a random number.

Compared with the Pedersen commitment scheme, the Hash commitment scheme enjoys higher efficiency but suffers weaker security. The computation overhead of Hash commitment scheme is more efficient than Pedersen commitment scheme. Instead, Hash commitment provides informal hiding property. Although an output of a hash function is thought to hide the underlying input in some works [34], the hiding property of hash function is not formally defined and guaranteed in the cryptographic literature.

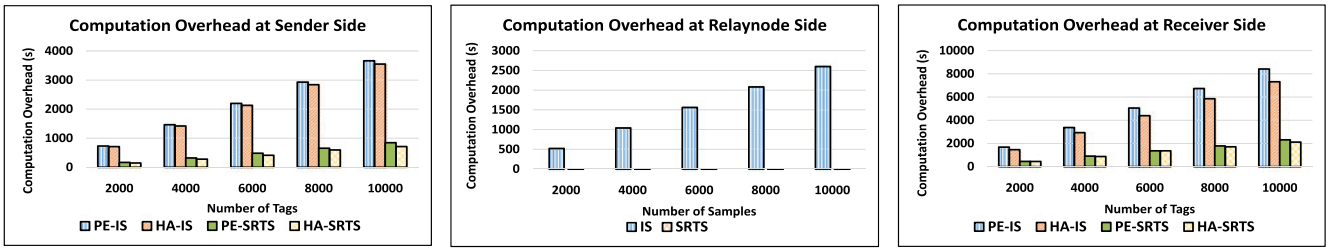
TABLE 3
Comparison of IS and SRTS in Terms of Cryptographic Operations

	Sender	Relaynode	Receiver
PE-IS	$n\texttt{Exp}_G$+$2n\texttt{Exp}_G$	$2n\texttt{Pair}_e$	$2n\texttt{Pair}_e$+$2n\texttt{Exp}_G$
PE-SRTS	$\texttt{Exp}_G$+$2n\texttt{Exp}_G$	$2\texttt{Pair}_e$	$2\texttt{Pair}_e$+$2n\texttt{Exp}_G$

6.2 Computation Overhead

We compare SRTS with the basic Item-level Solution (IS) as discussed in Section 2 in terms of cryptographic operations. According to our crypto selection, both IS and SRTS can be built on two compositions of digital signature and string commitment, namely "BLS signature + Perdersen commitment" and "BLS signature + Hash commitment". We term the two instantiations as PE-based scheme and HA-based scheme. We compare the performance of PE-IS, HA-IS and PE-SRTS, HA-SRTS. The computation overhead of BLS signature is dominated by exponentiation operation $\texttt{Exp}_G$ on group G and pairing operation $\texttt{Pair}_e$ of bilinear map e. The computation overhead of Pedersen commitment is dominated by exponentiation operation $\texttt{Exp}_G$ on group G. The computation overhead of Hash commitment is dominated by hash operation H.

Table 3 summarizes the cryptographic operations of PE-IS and PE-SRTS incurred at the three participants respectively, when a batch of n products are transferred through RTS system. The comparison of HA-IS and HA-SRTS is similar and is



(a) Computation overhead (s) at Sender side (b) Computation overhead (s) at Relaynode side (c) Computation overhead (s) at Receiver side

Fig. 7. Comparison of IS and SRTS.

ignored. The computation benefit of SRTS is derived from the signature processing overhead (signature generation for Sender and signature verification for Relaynode and Receiver). SRTS requires the three participants to process 1 signature while IS requires the three participants to process n signatures. As shown in Table 3, SRTS reduces computation complexity from $n\mathtt{Exp}_G$ to $\mathtt{Exp}_G$ at Sender side and from $2n\mathtt{Pair}_e$ to $2\mathtt{Pair}_e$ at both Relaynode and Receiver sides. Importantly, SRTS incurs constant computation complexity at Relaynode regardless of the size of the product batch, thus avoiding Relaynode as a bottleneck of the RTS system.

6.3 Tag Memory Overhead

We compare with the IS as a baseline in terms of tag storage overhead. Recall that SRTS requires each tag to store a crypto-ID, which is a commitment concatenated with an in-batch index. We set the in-batch index to be 16 bits long to support at most 2^{16}=65,536 products in a batch. Compared with IS, SRTS avoids the tag storage of digital signatures. PE-IS and HA-IS raise 672 bits storage while PE-SRTS and HA-SRTS raise 336 bits storage. We see that SRTS saves about 50 percent tag storage overhead compared with IS. Note that SRTS achieves more tag storage benefit if other long-length digital signature schemes are adopted.

6.4 Experiment Results

In our implementation, we adopt the Pairing Based Cryptography (PBC) libraries [35], [36], [37] to implement BLS signature, Pedersen commitment and Hash commitment. All the experiment results represent the average of 10 trials.

Each of Sender, Relaynode and Receiver has a backend server and multiple readers. The readers are used to bundle tag carried messages and forward the messages to the server via internal network for further computation. Current enterprises often own commercial-level servers to complete their computation tasks. In our experiments, we use a computation abundant PC to simulate the commercial-level server of each of the three participants, and conduct our experiments on the PC. The PC is equipped with a 16-Core AMD Opteron Processor and 16 GB RAM, running 64-bit Ubuntu 13.10.

We compare the performance of IS and SRTS at the three participants to transfer a batch of n products. In our experiment, we generate n random production messages. We vary n from 2,000 to 10,000. Figs. 7a, 7b, 7c show the experiment results at the three participants, respectively. The experiment results show that SRTS incurs far less computation overhead compared with IS. To transfer a batch of 10,000 products with PE-SRTS, Sender costs 845 s (3,662 s with PE-IS), Relaynode costs 0.257 s (2,601 s with PE-IS) and Receiver costs 2,307 s (8,416 s with PE-IS). Notice that at Relaynode side, SRTS incurs constant computation overhead regardless of the number of products. The reason is that Relaynode only needs to run PriVerify() algorithm once.

Table 4 lists the speed up ratio of SRTS against IS at the three participants. At Sender side, SRTS achieves 4.4 to 5.2 times speed up (lines 1-2). At Relaynode side, the speed up of SRTS grows linearly with the number of products, from 2,000 to 10,000 times (line 3). At Receiver side, SRTS achieves 3.2 to 3.8 times speed up (lines 4-5). The experiment results confirm our complexity analysis shown in Table 3.

6.5 Parallelization of SRTS

A design advantage of SRTS is that it is highly parallelizable. In transferring a batch of n products, the computation task of Sender can be divided into n independent commitment computation tasks and a signing task, and the computation task of Receiver can be divided into n independent commitment computation tasks and a signature verifying task.

We implement a parallel version of SRTS and compare the computation overhead of parallel-SRTS and original SRTS to process a batch of n products. We vary n from 2,000 to 10,000. Figs. 8a, and 8b show the experiment results at Sender and Receiver sides, respectively. Our results show that parallel-SRTS achieves obvious speed up compared with SRTS. To transfer a batch of 10,000 products with PE-parallel-SRTS, Sender costs 153 s (755 s with PE-SRTS) and Receiver costs 933 s (2,162 s with PE-SRTS).

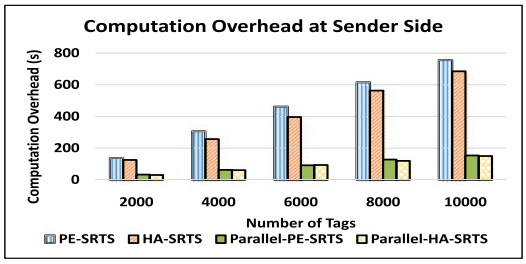
Table 5 lists the speed up ratio of parallel-SRTS and SRTS at Sender and Receiver sides. At Sender side, parallel-SRTS achieves 4 to 5 times speed up (lines 1-2). At Receiver side, parallel-SRTS achieves 2 to 2.3 times speed up (lines 3-4).

6.6 Tolerating Tag Errors and Failures

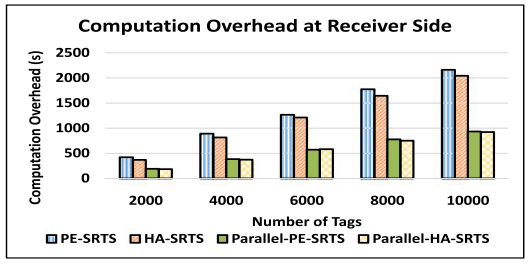
In SRTS, when a product batch is transferred in the RTS system, Relaynode and Receiver need to collect the crypto-IDs from the attached tags to recover a commitment batch, from which they can further derive a CB pair or an MB pair.

TABLE 4
Speed-Up Ratio of SRTS versus IS

Number of tags	2,000	4,000	6,000	8,000	10,000
Sender side-PE	4.3	4.5	4.5	4.4	4.3
Sender side-HA	4.7	5.0	5.1	4.7	4.9
Relaynode side	1,977	4,262	6,290	8,097	10,120
Receiver side-PE	3.7	3.7	3.7	3.7	3.6
Receiver side-HA	3.3	3.3	3.2	3.4	3.4



(a) Computation overhead (s) at Sender side



(b) Computation overhead (s) at Receiver side

Fig. 8. Comparison of SRTS and parallel-SRTS.

However, RFID tag is not a robust medium to carry message. In specific, a tag may suffer from: (1) tag error, i.e., the tag carries a wrong crypto-ID; (2) tag failure, i.e., the tag loses its functionality and its crypto-ID cannot be read anymore.

To tolerate tag errors and tag failures, we propose to extend the concept of crypto-ID to design redundant crypto-ID. Our idea is to combine Reed-Solomon code (RS code) with crypto-ID. A Reed-Solomon code is termed as RS(n, k) with s-bits symbols. The RS encoder takes k s-bits symbols to generate n redundant s-bits codewords. The RS decoder can correct up to s errors or up to r erasures with $2s + r < 2t$.

Recall that a crypto-ID is an indexed commitment $i||mc_i$. For a product batch with n products, we encode the corresponding commitment batch $\{mc_i\}_n = mc_1||mc_2||\ldots||mc_n$ into a redundant commitment batch by encoding every $k\times s$ bits of the commitment batch into $n\times s$ bits code words. We then equally divide the redundant commitment batch into n pieces. For each piece rmc_i, we concatenate it with an index i and generate a redundant crypto-ID $i||rmc_i$. Clearly, our redundant crypto-IDs can tolerant tag errors and tag failures.

We compare the performance of RS code with that of PE-SRTS. We use the implementation of a popular RS(255, 223) code [42]. We conduct two groups of experiments: (1) We use the PriSign() algorithm to generate a CB pair and then use the RS encoder to encode the commitment batch contained in the CB pair into a redundant commitment batch. (2) We use the RS decoder to decode a redundant commitment batch into a commitment batch, use the batch to form a CB pair, and run Check() algorithm on the CB pair. We measure the percentage of the time of the RS encoder/decoder against the total time and summarize the result in Table 6. From the table, we can see that the performance of RS code is negligible comparing with that of PE-SRTS. In the worst case, the overhead of RS code accounts for less than 0.2 percent of that of PE-SRTS.

TABLE 5
Speed-Up Ratio of Parallel-SRTS versus SRTS

Number of tags	2,000	4,000	6,000	8,000	10,000
Sender side-PE	4.1	4.8	5.0	4.8	4.9
Sender side-HA	4.0	4.2	4.2	4.6	4.5
Receiver side-PE	2.2	2.3	2.2	2.2	2.3
Receiver side-HA	1.9	2.1	2.0	2.1	2.2

TABLE 6
Comparison of RS Code with PE-SRTS (Seconds)

Tag num	PriSign	Encode	Per	Check	Decode	Per
2×10^3	144	0.263	0.2%	131	0.063	0.048%
4×10^3	304	0.427	0.1%	309	0.129	0.042%
6×10^3	457	0.545	0.1%	453	0.188	0.041%
8×10^3	565	0.759	0.1%	610	0.253	0.041%
10×10^3	722	0.926	0.1%	714	0.318	0.045%

6.7 Implementation on Commodity C1G2 RFID Systems

SRTS does not require any modifications to the commodity passive tags or implement additional cryptographic functionality on the tags. Instead, a tag only needs to carry short crypto-IDs for reading and writing purposes. SRTS satisfies the 512 bits-storage constraint of commodity passive tags. We use the Write command to write data into RFID tags. One Write command allows the reader to write a 16-bit data block. To write large-length data, the reader needs to divide the data into multiple 16-bit blocks and write them via several Write commands. On the other hand, The Read command supports bulk data collection, which allows the reader to collect up to 512 bits per Read operation.

We use the Alien ALR 9,900+ commodity RFID reader with regular parameters (e.g., 30 dBm transmission power) to interrogate commodity passive RFID tags. The data transfer program is developed based on the Alien RFID reader SDK codes. Our implementation only requires the C1G2 routine operations. So we believe that our implementation can be easily extended to other commodity RFID platforms. We select two different types of widely used passive tags – ALN-9640 and AD-224 tags both with 512-bit user memory.

We focus on the communication overhead of data transfer between the reader and the tag. Fig. 9 shows the communication overhead involved in the transfer of a 336-bits crypto-ID. As we can see, it requires more time to write a crypto-ID into tag, because as mentioned the Write command only allows the reader to write a 16-bit data block per Write operation. As a result, the reader needs to first divide the ID into several blocks and transfer them one by one which consumes longer time. In comparison, the Read

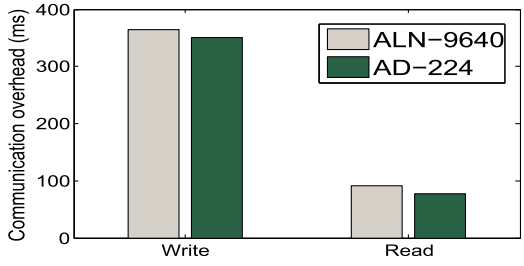


Fig. 9. Overhead (ms) of a 336-bits crypto-ID.

operation consumes less time since it only requires one Read operation to collect the whole ID.

7 Related Work

Currently, many works have studied security issues in RFID systems. Private preserving authentication (PPA) protocols [7], [8], [9], [10], [11], [12], [13], [14], [15] are such a type of protocols which enable a reader to authenticate the validity of a tag in a privacy-preserving way. In these protocols, the reader interacts with the tag in several rounds for authentication. The interacted messages are computed based on a secret key shared between the reader and the tag. If the authentication is successful, the reader can locate a record from a back-end database for the tag. The shared secret key can thus be treated as a crypto-ID for identification and index purposes. These works target at tag authentication problem and thus have totally different goals with our work.

Recently, some works have studied security issues in RFID-enabled supply chain systems [16], [17], [19]. Juels et al. [16] consider key distribution issue in RFID-enabled supply chain. Secure keys are directly stored in tags by using secret sharing. During the supply chain, only authorized distributors can recover the secure keys from tags and use them to decrypt the data stored in tags. Although key supply chain simplifies data privacy protection, some desired security properties (e.g., data non-repudiation) are not easy to be obtained. Blass et al. [17] propose a technique to authenticate whether a tag has been processed through a valid path in a supply chain network. The idea is to use polynomial signature together with homomorphic encryption, which allows the path information stored in a tag to be continuously updated when it flows through a valid path. Their technique provides authentication and privacy guarantee for the path information. However, the path information must be generated in fixed format (looks like random numbers), while we aim to provide security guarantee for general production messages. Chaves and Kerschbaum [19] propose a solution to protect privacy of production message in RFID-enabled product recall. In their solution, sensitive production message can be encoded in a privacy-preserving manner and stored in tags for problematic product identification. Their solution, however, does not consider data non-repudiation. Besides, our work targets at a different model of RFID-enabled supply chain systems comparing with all these works.

We design PVS scheme to provide security guarantees and reduce signature processing overhead, including both computation and storage, for large-scale RTS system. We notice that in the crypto literature, there are many other powerful signature schemes [38], [39], [40], [41]. We next discuss the suitability of these schemes when deploying in RTS system. Designated verifier signature [38] can convince a designated verifier the *authenticity* of a signed message. This scheme cannot be directly deployed in RTS system as it convinces a designated verifier in such a way that the verifier cannot prove the signature to a third party. In RTS system, however, sender needs to transfer *non-repudiable* messages to relaynode and receiver, so that the two can record these messages as evidences and later prove them to an authority for arbitration. Multi-signature [39], aggregate signature [40] and batch signature [41] are designed to reduce signature processing overhead. These schemes, however, fall in following drawbacks when deploying in RTS system. First, multi-signature and aggregate signature work in a multi-signers scenario, while in RTS system, all the signatures are signed by sender. Second, batch signature fasts signature verification overhead, but does not reduce signature storage overhead. Finally, all the three schemes do not protect privacy of the signed messages.

8 Conclusion

In this paper, we target at security and privacy issues in RFID supply chain systems. We consider RFID-enabled Third-party Supply chain (RTS) system. We analyze the essential structure of RTS system and identify three inherent requirements about production messages. We design a Secure RTS system called SRTS, which incorporates a Private Verifiable Signature scheme, to achieve the desired requirements. With SRTS, the production messages of a product batch are equipped with privacy and non-repudiation properties and can be efficiently transferred in the RTS system. In the future work, we plan to improve SRTS so that the receiver can directly recover production messages from tag carried crypto-IDs.

Acknowledgments

The authors acknowledge the support from Singapore MOE AcRF grant MOE2013-T1-002-005, NTU NAP grant M4080738.020, Hong Kong ECS (PolyU 252053/15E), National Natural Science Foundation of China (Grant No. 61472068), and the Project funded by China Postdoctoral Science Foundation (Grant No. 2014M550466).

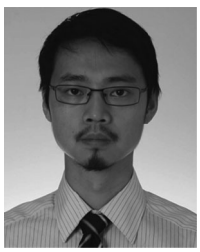
References

[1] R. Want, "An introduction to RFID technology," *IEEE Pervasive Comput.*, vol. 5, no. 1, pp. 25–33, Jan.-Mar. 2005.
[2] G. Vannucci, A. Bletsas, and D. Leigh, "A software-defined radio system for backscatter sensor networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2170–2179, Jun. 2008.
[3] Toll Global Logistics Expects RFID to Provide Significant Savings. (2010) [Online]. Available: http://www.rfidjournal.com/articles/view?7552.
[4] S. Cai, C. Su, Y. Li, R. Deng, and T. Li, "Protecting and restraining the third party in RFID-enabled 3PL supply chains," in *Proc. 6th Int. Conf. Inf. Syst. Security*, 2010, pp. 246–260.
[5] China e-commerce complaints and rights of public service platform. (2011) [Online]. Available: http://b2b.toocle.com/zt/315/list–11–1.html
[6] B. Santos and L. Smith, "RFID in the supply chain: Panacea or Pandora's box?" *Commun. ACM*, vol. 51, no. 10, 2008.
[7] L. Lu, J. Han, R. Xiao, and Y. Liu, "ACTION: Breaking the privacy barrier for RFID systems," in *Proc. IEEE INFOCOM*, 2009, pp. 1953–1961.
[8] T. Li, W. Luo, Z. Mo, and S. Chen, "Privacy-preserving RFID authentication based on cryptographical encoding," in *Proc. IEEE INFOCOM*, 2012, pp. 2174–2182.
[9] T. Li and R. Deng, "Vulnerability analysis of EMAP—an efficient RFID mutual authentication protocol," in *Proc. 7th Int. Conf. Availability, Rel. Security*, 2007, pp. 238–245.
[10] A. X. Liu, L. A. Bailey, and A. H. Krishnamurthy, "RFIDGuard: A lightweight privacy and authentication protocol for passive RFID tags," *J. Security Commun. Netw.*, vol. 3, no. 5, pp. 384–393, 2010.
[11] T. Van Le, M. Burmester, and B. de Medeiros, "Universally composable and forward-secure RFID authentication and authenticated key exchange," in *Proc. 2nd ACM Symp. Inf. Comput. Commun. Security*, 2007, pp. 242–252.

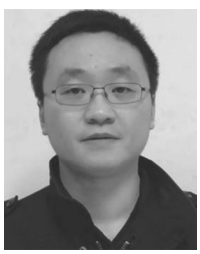
[12] M. Burmester, B. De Medeiros, and R. Motta, "Robust, anonymous RFID authentication with constant key-lookup," in *Proc. ACM Symp. Inf. Comput. Commun. Security*, 2008, pp. 283–291.
[13] A. Juels and S. Weis, "Defining strong privacy for RFID," in *Proc. IEEE PerCom, Workshop PerTec*, 2007, pp. 342–347.
[14] T. Dimitriou, "A secure and efficient RFID protocol that could make big brother (partially) obsolete," in *Proc. IEEE 4th Annu. IEEE Int. Conf. Pervasive Comput. Commun.*, 2006, pp. 269–275.
[15] C. C. Tan, B. Sheng, and Q. Li, "Secure and serverless RFID authentication and search protocols," *IEEE Trans. Wireless Commun.*, vol. 7, no. 3, pp. 1400–1407, Apr. 2008.
[16] A. Juels, R. Pappu, and B. Parno, "Unidirectional key distribution across time and space with applications to RFID security," in *Proc. USENIX Security*, 2008, pp. 75–90.
[17] E. Blass, K. Elkhiyaoui, and R. Molva, "Tracker: Security and privacy for RFID-based supply chains," in *Proc. 18th Annu. Netw. Distrib. Syst. Security Symp.*, 2011.
[18] K. Elkhiyaoui, E. Blass, and R. Molva, "CHECKER: On-site checking in RFID-based supply chains," in *Proc. 5th ACM Conf. Security Privacy Wireless Mobile Netw.*, 2012, pp. 173–184.
[19] L. W. F. Chaves and F. Kerschbaum, "Industrial privacy in RFID-based batch recalls," in *Proc. 12th Enterprise Distrib. Object Comput. Conf. Workshops*, 2008, pp. 192–198.
[20] E. Liu and A. Kumar, "Leveraging information sharing to increase supply chain configurability," in *Proc. Int. Conf. Inform. Syst.*, 2003, pp. 523–536.
[21] A. Melski, L. Thoroe, and M. Schumann, "Managing RFID data in supply chains," *Int. J. Internet Protocol Technol.*, vol. 2, no. 3/4, pp. 176–189, Dec. 2007.
[22] A. Hofer, A. Knemeyer, and P. Murphy, "The roles of procedural and distributive justice in logistics outsourcing relationships," *J. Bus. Logistics*, vol. 33, no. 3, pp. 196–209, 2012.
[23] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Adv. Cryptol.*, 1989, pp. 239–252.
[24] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. 17th Annu. ACM Symp. Theory Comput.*, 1985, pp. 291–304.
[25] C. Schnorr,"Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.
[26] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. 11th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 1991, pp. 129–140.
[27] G. Fuchsbauer, "Automorphic signatures in bilinear groups and an application to round-optimal blind signatures," in *Proc. IACR*, 2009.
[28] (2012). Alien Technology [Online]. Available: http://www.alientechnology.com.
[29] (2012). Class 1 Generation 2 UHF Interface Protocol Standard 'Gen2', EPCglobal [Online]. Available: http://www.epcglobalinc.org/standards/uhfc1g2.
[30] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," in *Proc. IEEE INFOCOM*, 2011, pp. 3101–3109.
[31] T. Li, S. Chen, and Y. Ling, "Identifying the missing tags in a large RFID system," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2010, pp. 1–10.
[32] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," in *Proc. 19th IEEE Int. Conf. Netw. Protocols*, 2011, pp. 363–372.
[33] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, pp. 297–319, 2004.
[34] M. Farb, Y. H. Lin, H. J. Kim, J. McCune, and A. Perrig, "SafeSlinger: Easy-to-use and secure public-key exchange," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.* 2013, pp. 417–428.
[35] B. Lynn. The pbc library. (2006) [Online]. Available: http://crypto.stanford.edu/pbc/.
[36] A. De Caro and V. Iovino. (2011). "jPBC: Java Pairing Based Cryptography," in *Proc. 16th IEEE Symp. Comput. Commun., ISCC 2011*, Kerkyra, Corfu, Greece, [Online]. Available: http://gas.dia.unisa.it/projects/jpbc
[37] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proc. IEEE Symp. Comput. Commun.*, 2011, pp. 850–855.
[38] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Proc. 15th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 1996, pp. 143–154.
[39] T. Okamoto, "A digital multisignature scheme using bijective public-key cryptosystems," *ACM Trans. Comput. Syst.*, vol. 6, pp. 432–441, 1998.
[40] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. 22nd Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 416–432.
[41] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1998, pp. 236–250.
[42] RS code implementation [Online]. Available: https://github.com/zxing/zxing.

Saiyu Qi received the BS degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2008, and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2014. He is currently an assistant professor with the School of Cyber Engineering, Xidian University, China. His research interests include applied cryptography, cloud security, distributed systems, and pervasive computing.

Yuanqing Zheng received the BS degree in electrical engineering, the ME degree in communication and information system from Beijing Normal University, Beijing, China, in 2007 and 2010, respectively, and the PhD degree in Nanyang Technological University from the School of Computer Engineering in 2014. He is currently an assistant professor with the Department of Computing in the Hong Kong Polytechnic University. His research interest includes human centered computing, mobile and wireless computing, RFID systems, etc. He is a member of the IEEE and ACM.



Mo Li received the BS degree in computer science and technology from Tsinghua University, Beijing, China, in 2004 and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2009. He is currently an assistant professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include wireless sensor networking, pervasive computing, and mobile and wireless computing. He is a member of the Association for Computing Machinery (ACM). He received the ACM Hong Kong Chapter Prof. Francis Chin Research Award in 2009 and the Hong Kong ICT Award Best Innovation and Research Grand Award in 2007.



Li Lu received the PhD degree from the Key Lab of Information Security, Chinese Academy of Science, in 2007. He is an associate professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include RFID technology and system, wireless network and network security. He is a member of the IEEE Communication Society and ACM.

Yunhao Liu received the BS degree in automation from Tsinghua University, China, in 1995, the MS and PhD degrees in computer science and engineering from Michigan State University, in 2003 and 2004, respectively. He is now Cheung Kong professor and dean of the School of Software at Tsinghua University, China. His research interests include RFID and sensor network, the internet and cloud computing. He is a fellow of the IEEE and ACM.