COLLECTOR: A Secure RFID-Enabled Batch Recall Protocol

Saiyu Qi*†, Yuanqing Zheng†, Mo Li†, Li Lu‡ and Yunhao Liu*§
*Department of Computer Science and Engineering, HKUST, Hong Kong
†School of Computer Engineering, Nanyang Technological University, Singapore
‡School of Computer Science and Engineering, University of Electronic Science and Technology, China
§TNLIST, School of Software, Tsinghua University, China

***Abstract*—Batch recall is a practically important problem for most industry manufacturers. The batches of products which contain flawed parts need to be recalled by manufacturers in time to prevent further economic and health loss. Accurate batch recall could be a challenging issue as flawed parts may have already been integrated into a large number of products and distributed to customers. The recent development of Radio Frequency Identification (RFID) provides us a promising opportunity to implement batch recall in an accurate and efficient way. RFID-enabled batch recall provides us the opportunity to further enhance the security of batch recall operation, allowing us to achieve recognition of problematic products, privacy preserving of production pattern, recall authentication and non-repudiation, etc. In this paper, we thoroughly study the security aspects and identify the unique requirements in RFID-enabled batch recall. We propose a practically secure protocol, COLLECTOR, to enable accurate, secure and efficient RFID batch recall.**

***Index Terms*—RFID; batch recall; security**

I. Introduction

Batch recall is a practically important problem for most industry manufacturers. The batches of products containing flawed parts, which may have already been sold to a large number of customers, need to be recalled in time to prevent economic and health loss. The selling of problematic products is considered as major threats to human beings. In medicine and food industries, incidents of contamination may even cause the loss of human life. In some countries, the government directly participates in batch recall and enforces the manufacturers to recall the problematic products. For example, in the U.S., batch recall in the food industry is enforced by the Food and Drug Administration (FDA).

In modern industry process, the product manufacturing often involves multiple manufacturers and the end-products contain different parts supplied from different manufacturers. In a general manufacturing process, the intermediate batches of product parts flow a series of intermediate manufacturers and finally reach an end manufacturer. At each step, an intermediate manufacturer receives intermediate batches from its upstream partners, builds its own parts, generates new intermediate batches and supplies to downstream partners. The end manufacturer finally outputs batches of end-products. Fig. 1 illustrates such a manufacturing process where four intermediate manufacturers: t_1, t_2, t_3, t_4 and an end manufacturer T participate in the production chain. Each manufacturer builds its own parts on received intermediate batches and produces new batches. As depicted in Fig. 1, t_3 and t_4 receive intermediate batches from t_1 and t_2, and produce new intermediate batches to T. During the production process, the number of produced intermediate batches may vary across different manufacturers. The current manufacturer may use different intermediate batches from upstream partners to produce one output batch (e.g., t_3 uses batch 1 and batch 2 to batch 5), yet one intermediate batch may be used to produce multiple output batches (e.g., T uses batch 7 for batch 9, batch 10, batch 11 and batch 12). When a problematic end-product in the market is detected, it will be reported to the end manufacturer *T*. After re-checking, an intermediate batch of flawed parts contributed to the end-product can be traced. The objective of batch recall is to identify and recall all batches of end-products that contributed by that particular intermediate batch.

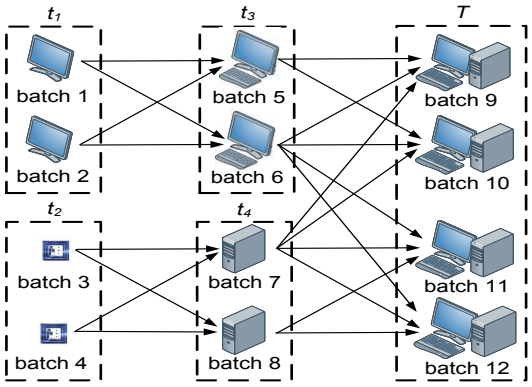


Fig. 1. The manufacturing process of modern industry.

The state-of-the-practice solution to batch recall is usually to maintain the entire production messages on the production chain and mark main components of an product with part numbers. When a recall event happens, the part numbers of potentially affected components are announced to public and the customers will have to decompose the end-products for verification (e.g., Lenovo battery recall [1]), which is neither accurate nor efficient. An alternative scheme is to recall the entire batches of end-products manufactured during an entire

period (e.g., Toyota automobile recall 2009-2011 [2]), which however, incurs huge extra overhead and unnecessary loss to the manufacturers.

The recent advances in Radio Frequency Identification (RFID) technology provide promising solutions to save the batch recall operation from such a miserable situation. Small RFID tags can be attached to end-products, and the recall messages, which contain batch-related information, can be thus stored and later retrieved. With a reported problematic end-product, the end manufacturer uses its recall message to recover the production messages of intermediate batches contributed to the product and traces the problematic intermediate batch. When the batch is identified, the end manufacturer announces the batch information corresponding to the identified batch to public for identification purpose. Apparently, the use of RFID facilities the trace of problematic intermediate batch and enables item-level identification. The successful deployment of RFID technology further enables us to provide substantial security for the batch recall operation.

In this paper, we systematically study the requirements for RFID-enabled batch recall. We consider manufacturers and customers in the scenario where end-products are distributed from manufacturers to customers and problematic end-products are recalled from customers to manufacturers. The most important two functional requirements are traceability—the end manufacturer could use the recall message of the reported problematic end-product to recover the production messages of all the intermediate batches contributed to the product, and accuracy—the end manufacturer could publish identification message and rely on customers to check and return problematic end-products based on the tag carried recall messages. Secondly, we concern the privacy of the recall messages in RFID tags, and try to protect them from being used by malicious entities to infer the production pattern of the production chain. We also consider two authentication requirements: recall authentication and recall non-repudiation between the end manufacturer and customers. When an end-product is identified as problematic, the customer can acquire a recall-evidence about the product for non-repudiation purpose (recall non-repudiation). On the other hand, the end manufacturer expects faithful receipt of returned problematic end-product for authentication purpose (recall authentication). With RFID technique, we want to design recall messages that can serve as electronic receipts to provide the two requirements. Finally, we concern several critical efficiency requirements to design viable RFID batch recall protocol.

Our contribution can be summarized as follows. First, we define the system model to formulate the batch recall problem. According to this model, we identify function requirements, security requirements and efficiency requirements in RFID-enabled batch recall. Based on these requirements, we propose COLLECTOR, a distributed secure batch recall protocol. Following COLLECTOR, the end manufacturer can recover correct production messages for any reported problematic end-product by using its recall message and the customers can accurately identify the problematic end-products by using the tag carried recall messages as well as an identification message published by the end manufacturer. At the same time, the recall messages are well formed and do not leak the production pattern of the production chain. COLLECTOR also guarantees the hardness in recovering witness messages from the recall messages of good end-products, providing authentication and non-repudiation. Finally, COLLECTOR is built on prevalent cryptographic primitives to satisfy the desired efficiency requirements. We do thorough analysis and show that the proposed protocol meets all identified requirements.

The rest of this paper is organized as follows. We describe the batch recall scenario and discuss the requirements of RFID-enabled batch recall in Section II. In Section III, we briefly describe the design principle of COLLECTOR. In Section IV, we present the design details of COLLECTOR. In Section V, we analyze the requirements achieved by COLLECTOR. In Section VI, we examine the efficiency of COLLECTOR on current commodity tags. At last, we review the related works in Section VII and conclude this paper in Section VIII.

II. PRELIMINARY

A. Batch recall scenario

We consider three types of entities: intermediate manufacturer t_i, end manufacturer T and customer C_i in our system model. An end manufacturer T relies on a set of intermediate manufacturers $\{t_i\}$ to produce end-products and sells them to customers. A customer C_i can be a dealer or an individual. Dealers directly buy end-products from T and sell them to individuals. All the customers holding the end-products form a customer set $\{C_i\}$, which is huge and unpredictable from the perspective of T. We divide the batch recall scenario into three processes: production process, distribution process and recall process.

Production process: The production process consists of N sequential production steps: $\{t_i\}^1 \rightarrow \{t_i\}^2 \rightarrow \cdots \rightarrow \{t_i\}^N$. Each step j ($1 \leq j \leq N$) involves a subset of intermediate manufacturers $\{t_i\}^j$. In a particular step j, each $t_i \in \{t_i\}^j$ receives intermediate batches from one or multiple $t_i \in \{t_i\}^{j-1}$, produces its own intermediate batches and sends them to one or multiple $t_i \in \{t_i\}^{j+1}$. Finally, T receives intermediate batches from each $t_i \in \{t_i\}^N$ and produces batches of end-products. As a result, each end-product is contributed by multiple intermediate batches.

As an example, Fig. 1 shows a production process with two production steps. Step 1 involves $\{t_1, t_2\}$ and step 2 involves $\{t_3, t_4\}$. After the production, each end-product of batch 12 is contributed by intermediate batches 1, 2, 3, 4, 6, 7, 8.

Distribution process: In the distribution process, end-products are sold from the end manufacturer T to the customer set $\{C_i\}$. Specifically, dealers directly buy end-products from T and then sell them to individuals. Note that a C_i may not buy one or multiple complete batches of end-products. Instead, the purchased end-products may come across multiple batches.

Recall process: Later, if an end-product is found to be problematic by a C_i, the C_i will report this event to T and return the end-product. T then cooperates with the intermediate manufacturer set $\{t_i\}$ to trace the problematic source.

If an intermediate batch supplied by a t_i is located to be problematic, we say a recall event happens. T will then collect all the problematic end-products contributed by this intermediate batch from $\{C_i\}$.

B. Integration with RFID system

A typical RFID system consists of a reader and a set of tags. A tag is a tiny transmitter with constrained storage and computation power. A tag can be attached to an object and store some data which describes the object. On the other hand, an RFID reader is used to read and write tag carried data via wireless channels. Compared with tags, readers can be equipped with more storage and computation power.

In RFID-enabled batch recall scenario, each end-product is attached with a tag for identification and a customer C_i should be able to read/write tag carried data. For a dealer, we assume that it is equipped with readers to read the attached tags. For an individual, we assume that it can approach the dealer from which it buys the end-products and requests the dealer to read/write the attached tags. We envision that in the near future, personal devices such as smart phones could be equipped with scanning capability so that individuals can complete the read/write tasks by themselves.

C. Requirement analysis

The practical deployment of RFID system is subject to various requirements. To accurately identify problematic end-products, certain recall messages should be stored in tags for identification. For example, a straightforward approach is to assign a batch number for each intermediate batch and store proper batch number sets into tags. Later, *T* publishes the batch number of the problematic intermediate batch for customers to identify. Such a solution, however, does not provide any security protection. For example, a business competitor can easily scan the tag carried batch numbers to infer the production pattern of the production chain. In specific, the competitor can immediately infer (1) how many intermediate batches are used to produce an end-product (by counting the number of a batch number set), (2) if two end-products are contributed by same intermediate batch(es) (by comparing two batch number sets). We define the above two types of information as *non-trivial information* of production pattern.

Instead, we aim to provide security protections to enforce the correct execution of batch recall operation while achieving function requirements and efficiency requirements. We list all the desired requirements as follows.

1) Function requirements: We identify two functional advantages that an RFID-enabled batch recall protocol could provide.

• **Traceability:** When receiving a reported problematic end-product, T should be able to use its recall message to recover the production messages of the intermediate batches contributed to the end-product, which enables T to trace the problematic intermediate batch. Traceability is required by T.

In COLLECTOR, each intermediate manufacturer $t_i \in \{t_i\}$ creates production records to store the production messages of its supplied intermediate batches. Also, T creates batch records to store the state of its batches of end-products and recall messages for each end-product. The production records, batch records and recall messages are correlated so that *T* could link each recall message to a proper set of production records.

• **Accuracy:** When a recall event happens, T should be able to publish an identification message so that all the problematic end-products can be precisely identified by the customer set $\{C_i\}$. Accuracy is required by T and $\{C_i\}$.

COLLECTOR leverages the correlation between production records, batch records and recall messages for *T* to generate a proper identification message.

2) Security requirements: An RFID-enabled batch recall protocol needs to satisfy a number of security requirements to prevent potential malicious behaviors.

• **Production privacy:** Recall messages of end-products should not leak *non-trivial information* about the production pattern of the production chain. Production privacy is required by T.

Different with the straightforward approach, in COLLECTOR, T generates recall messages without the knowledge of batch numbers of intermediate batches to prevent the leakage of production pattern.

• **Recall authentication:** When T receives an end-product from a C_i, T should be able to authenticate that this end-product is truly problematic. Recall authentication is required by T.

In COLLECTOR, T encrypts a unique witness message into each recall message. COLLECTOR ensures that a C_i can only recover witness messages from the recall messages of problematic end-products. Therefore, T requires a C_i to provide a recovered witness message for each returned end-product for authentication.

• **Recall non-repudiation:** When a C_i identifies an end-product as problematic, it should be able to acquire a recall-evidence about the product so that T cannot refuse to recall it. Recall non-repudiation is required by $\{C_i\}$.

In COLLECTOR, as a C_i can only recover a witness message from the recall message of an problematic end-product, the C_i can record the recovered witness message and the recall message as a recall-evidence. A key point here is that the C_i must convince an authority that the recall-evidence is acquired after identifying a problematic end-product, but not faked by itself. COLLECTOR resorts to signature scheme to solve this problem.

3) Efficiency requirements: A viable RFID-enabled batch recall protocol should be highly efficient. We identify several critical efficiency requirements toward this goal.

• **Key management:** The design of an RFID-enabled batch recall protocol should not rely on the customer set $\{C_i\}$ to securely store secret keys. Considering the varying skills,

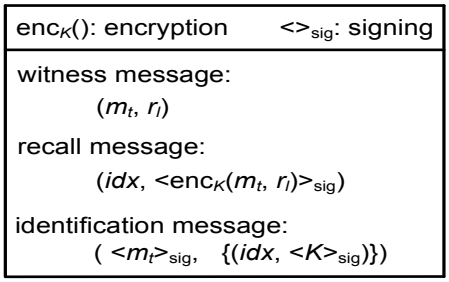


Fig. 2. The format of the three types of messages.

interests and cultures of a huge quantity of customers, their stored keys may be easily leaked or compromised.

COLLECTOR adopts a centralized key management manner—only T needs to securely store secret keys, and thus simplifies the complexity and enhances security.

• **Computation:** The computation overhead of an RFID-enabled batch recall protocol should be lightweight to support large scale recall operation involving a huge quantity of customers.

COLLECTOR builds on two prevalent cryptographic primitives: symmetric encryption scheme and signature scheme, which can be easily deployed and fast executed. Moreover, COLLECTOR uses a type of message called batch index, which enables a C_i to fast filter good end-products with no computation overhead. This means that COLLECTOR is especially efficient when only a small fraction of end-products are problematic.

• **Tag overhead:** The usage of RFID technique raises efficiency concern about RFID tags. A typical tag has a small memory space and supports extremely limited computational operations.

In COLLECTOR, during the life cycle of an end-product, the attached tag only needs to store a recall message and a witness message (optional). Both types of messages have constant size and suit the storage constraint (512 bits) of current commercial RFID tags.

III. PROTOCOL OVERVIEW

In this section, we describe the design principle of COLLECTOR.

A. Principle of COLLECTOR

To simplify the description, we use a notion $<m>_{sig}$ to denote a signed message m generated by T. Therefore, $<m>_{sig}$ consists of a message m and a signature signed on m.

In COLLECTOR, each intermediate manufacturer $t_i \in \{t_i\}$ creates production records for its supplied intermediate batches. A production record contains a production message about the corresponding batch and is indexed by a batch number b_j.

T cooperates with $\{t_i\}$ to track which intermediate batches are contributed to its batches of end-products and creates batch records for its batches. A batch record corresponds to a batch of end-products and contains a batch index *idx*, a secret key K and a batch number set $\{b_j\}$. T also generates witness messages and recall messages for each end-product of the corresponding batch. A witness message (m_t, r_l) contains a fixed recall token m_t and a random number r_l. A recall message (*idx*, $<CT>_{sig}$) contains the batch index *idx* and a signed ciphertext $<CT>_{sig}$, where CT encrypts a witness message by using K. T then loads the recall messages into the tags of the end-products. As recall messages contain signatures of T, they can be used as buying-evidences for customers to store.

Upon receiving a reported problematic end-product, T links its recall message to a batch record through batch index, and links the batch record to a set of production records through batch numbers. With the linked production records, T then cooperates with $\{t_i\}$ to trace the number of the problematic intermediate batch.

With the traced batch number, T initializes an identification message as $(<m_t>_{sig})$. Recall that a batch record contains a batch index *idx*, a secret key K and a batch number set $\{b_j\}$. If $\{b_j\}$ contains the problematic batch number, T adds the pair (*idx*, $<K>_{sig}$) into the identification message. After checking each batch record, T publishes the identification message.

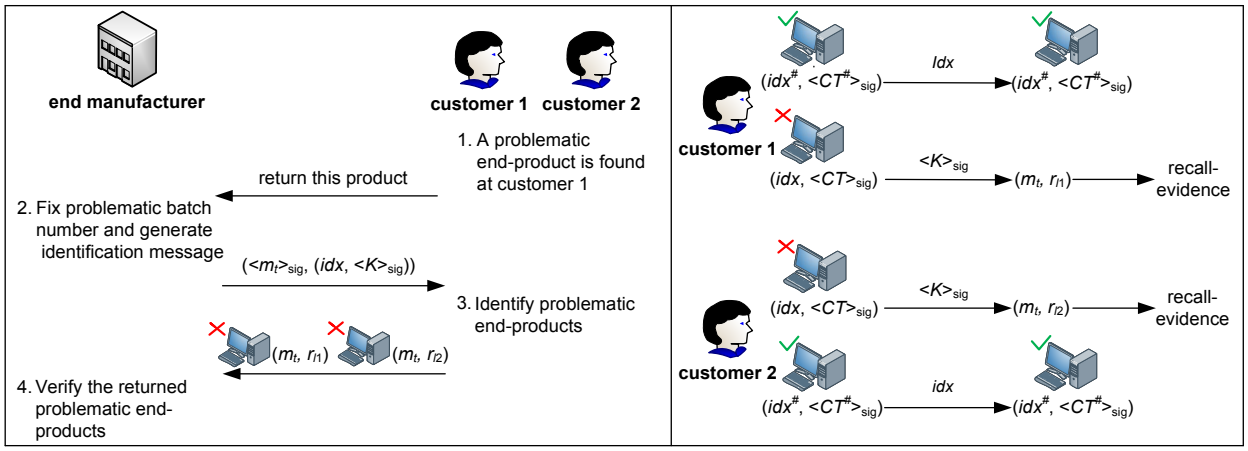
On the other hand, a customer C_i identifies an end-product as problematic/good by deciding if its recall message (*idx*, $<CT>_{sig}$) can be linked to a pair (*idx*, $<K>_{sig}$) of the identification message through *idx*. If linked (problematic), the C_i decrypts $<CT>_{sig}$ to a witness message (m_t, r_l) by using $<K>_{sig}$ of the linked pair. The C_i then loads (m_t, r_l) into the attached tag before returning the end-product. If not linked (good), the C_i cannot do so. Moreover, as each witness message contains a random number, the C_i cannot guess the encrypted witness message. Therefore, if a C_i can provide a correct witness message for a returned end-product, T convinces that the product is truly problematic.

When a C_i identifies an end-product as problematic, the C_i acquires a signed tuple $(<CT>_{sig}, <K>_{sig}, <m_t>_{sig})$ satisfying decryption relation. $<m_t>_{sig}$ is fetched from the identification message and is used to show that $<CT>_{sig}$ can indeed be decrypted by $<K>_{sig}$. The C_i records this tuple as a recall-evidence. Note that the tuple needs to be signed to convince an authority that the tuple is acquired after identifying a problematic end-product, but not faked by the C_i.

We summarize the format of witness message, recall message and identification message in Fig. 2.

B. An illustrative example

We use an example to illustrate the basic work flow of COLLECTOR. As shown in Fig. 3(a), when receiving a reported problematic end-product, the end manufacturer traces the batch number of the problematic intermediate batch. It then finds that a batch record contains the problematic number. The end manufacturer generates an identification message $(m_t, (idx, <K>_{sig}))$ based on the affected batch record and publishes the identification message. As shown in Fig. 3(b), each customer has a good end-product and a problematic end-product. The customers directly filter the good ones as their recall messages cannot be linked to (*idx*, $<K>_{sig}$) through *idx*. The customers



(a) Communication between end manufacturer and customers

(b) Identification of problematic end-products at customer side

Fig. 3. Overview of COLLECTOR.

also recover witness massages from the recall messages of the problematic ones by using $<K>_{\text{sig}}$. After that, the customers acquire recall-evidences for the problematic ones. Finally, as shown in Fig. 3(a), the customers return their problematic end-products as well as the recovered witness messages for the end manufacturer to authenticate.

IV. PROTOCOL DESIGN

In this section, we present the design of COLLECTOR. COLLECTOR consists of four phases: production phase, distribution phase, tracking phase and recall phase.

Initially, T generates a signing-verification key pair of a signature scheme. T applies a certificate for the verification key from a key authority and publishes the verification key/certificate pair. As a result, anyone can verify the validity of the verification key and use the key to verify the validity of signed messages generated by T.

A. Production phase

The production phase involves the production process $\{t_i\}^1 \rightarrow \{t_i\}^2 \rightarrow \cdots \rightarrow \{t_i\}^N$. During the process, T cooperates with the intermediate manufacturer set $\{t_i\}$ to track batch numbers of intermediate batches, creates batch records for its batches of end-products and generates recall messages for its end-products.

P1. During the production process, each intermediate manufacturer $t_i \in \{t_i\}$ creates a production record for each of its supplied intermediate batches in its database. A production record (b_j, PM_{b_j}) contains a batch number b_j and a production message PM_{b_j} about the corresponding batch.

The end manufacturer T cooperates with the intermediate manufacturer set $\{t_i\}$ to track which intermediate batches are contributed to its batches of end-products. Through cooperation, T collects a batch number set for each of its batches. A batch number set contains the batch numbers of all the intermediate batches contributed to the corresponding batch.

After the production process, T attaches its batches of end-products with tags in item-level.

P2. T generates a recall message for each of its end-products and loads the message into the attached tag.

For each batch of end-products with batch number set $\{b_j\}$, T chooses a batch index *idx* and a secret key K of a symmetric encryption scheme. For each end-product in the batch, T encrypts a witness message (m_t, r_l) into a symmetric-ciphertext CT by using K, where r_l is a random number. T then generates a recall message:

$$(idx, <CT>_{\text{sig}})$$

and loads the recall message into the attached tag. After processing each end-product of the batch, T creates a batch record:

$$(idx, K, \{b_j\}, \{(m_t, r_l)\})$$

for the batch in its database. $\{(m_t, r_l)\}$ is the set of witness messages generated for all the end-products of the batch.

B. Distribution phase

In this phase, T distributes end-products to the customer set $\{C_i\}$. For the reason of brevity and clarity, we only focus on the distribution from T to dealers. The further distribution from dealers to individuals is similar and is ignored.

In a trading transaction between T and a customer C_i, T directly transfers the required end-products to the C_i. For each received end-product, the C_i reads the recall message from the attached tag and verifies if its contained symmetric-ciphertext CT is correctly signed by T. If yes, the C_i accepts the end-product. As the contained symmetric-ciphertext is correctly signed, the recall message serves as a buying-evidence of the end-product and the C_i can choose to locally store the recall

message. The transaction ends after the C_i has accepted all the received end-products.

C. Tracking phase

Tracking phase is event-driven: when receiving a reported problematic end-product from a customer C_i, T cooperates with the intermediate manufacturer set $\{t_i\}$ to trace the batch number of the problematic intermediate batch contributed to this product.

T1. T reads the recall message from the tag of the reported problematic end-product. T then uses the batch index *idx* of the recall message to search a batch record (*idx*, $\{b_j\}$, $\{(m_t, r_l)\}$) in its database with the same *idx*. If such a record is found, T fetches the batch number set $\{b_j\}$ of the record.

T2. T recovers the production messages of intermediate batches corresponding to $\{b_j\}$. Recall that these messages are stored in the databases of the intermediate manufacturer set $\{t_i\}$ (in the form of production record (b_j, PM_{b_j})). As a result, T can easily fetch these production messages by using $\{b_j\}$ as indexes to query the databases of $\{t_i\}$.

T3. After querying, T knows which intermediate batches are contributed to the problematic end-product as well as their production messages. T then cooperates with $\{t_i\}$ to trace the batch number of the problematic intermediate batch.

D. Recall phase

With the traced batch number of the problematic intermediate batch, T then publishes an identification message for the customer set $\{C_i\}$ to identify and return problematic end-products. During this process, each C_i acquires a recall-evidence for each of its problematic end-products and T verifies if each returned end-product is truly problematic before accepting it.

R1. T initializes an identification message as $(<m_t>_{\mathsf{sig}})$. Suppose the traced problematic batch number is b_j^*, for each of its batch records (*idx*, *K*, $\{b_j\}$, $\{(m_t, r_l)\}$), T checks if b_j^* is contained in the batch number set $\{b_j\}$. If yes, T fetches the pair (*idx*, *K*), signs *K* and adds (*idx*, $<K>_{\mathsf{sig}}$) into the identification message. After checking all the batch records, T publishes the identification message:

$$(<m_t>_{\mathsf{sig}}, \{(idx, <K>_{\mathsf{sig}})\})$$

R2. When a customer C_i discovers the identification message, the C_i starts to identify if some of its end-products are problematic. For each end-product, the C_i reads the recall message (*idx*, $<CT>_{\mathsf{sig}}$) from the attached tag and uses the recall message as well as the identification message to identify if the product is problematic. The detail is summarized in **Algorithm 1**. If **Algorithm 1** outputs ($<K>_{\mathsf{sig}}$, $<m_t>_{\mathsf{sig}}$, (m'_t, r'_l)), the C_i verifies if $<K>_{\mathsf{sig}}$ and $<m_t>_{\mathsf{sig}}$ are correctly signed by *T*. If yes, the C_i identifies the end-product as problematic and creates a recall-evidence:

$$(<CT>_{\mathsf{sig}}, <K>_{\mathsf{sig}}, <m_t>_{\mathsf{sig}})$$

for the end-product. The C_i also writes the witness message (m'_t, r'_l) into the attached tag and returns the end-product to *T*.

Algorithm 1 Identification of problematic end-product

Input:
recall message: (*idx*, $<CT>_{\mathsf{sig}}$)
identification message: ($<m_t>_{\mathsf{sig}}$, $\{(idx, <K>_{\mathsf{sig}})\}$)

Procedure:
use *idx* $\in$ recall message to search a pair (*idx*, $<K>_{\mathsf{sig}}$) $\in$ identification message with the same *idx*;
if a pair (*idx*, $<K>_{\mathsf{sig}}$) is searched **then**
  fetch $<K>_{\mathsf{sig}}$ from (*idx*, $<K>_{\mathsf{sig}}$);
  fetch $<CT>_{\mathsf{sig}}$ from recall message;
  fetch $<m_t>_{\mathsf{sig}}$ from identification message;
  decrypt *CT* to (m'_t, r'_l) by using *K*;
  if $m'_t = m_t$ **then**
    output ($<K>_{\mathsf{sig}}$, $<m_t>_{\mathsf{sig}}$, (m'_t, r'_l));
  else output 'fault';
else output 'good'.

R3. For each returned end-product, T reads the witness message (m'_t, r'_l) from the attached tag and checks if (m'_t, r'_l) is contained in the witness message set $\{(m_t, r_l)\}$ of a certain batch record. If yes, T accepts the end-product as problematic and removes (m'_t, r'_l) from the witness message set.

V. Requirement analysis

In this section, we show that COLLECTOR achieves the function requirements, security requirements and efficiency requirements posed by RFID-enabled batch recall.

A. Function requirement analysis

Traceability: When receiving a reported problematic end-product, *T* should be able to recover the production messages of the intermediate batches contributed to the product. In COLLECTOR, *T* could link each end-product to the corresponding production messages as follows. Given an arbitrary end-product, *T* reads its recall message from the attached tag. *T* then uses the batch index *idx* of the recall message to search a batch record in its database indexed by the same *idx*. The batch number set $\{b_j\}$ of the searched batch record contains the batch numbers of all the intermediate batches contributed to the product. *T* finally uses these batch numbers to search production records in the databases of the intermediate manufacturer set $\{t_i\}$ indexed by the same batch numbers. The searched production records contain the production messages of all the intermediate batches contributed to the product.

Accuracy: With a traced problematic intermediate batch, *T* should be able to enable $\{C_i\}$ to identify all the end-products contributed by the batch. In COLLECTOR, *T* could link each intermediate batch with all the end-products contributed by the batch as follows. Given the batch number of an arbitrary intermediate batch, *T* searches the batch records whose batch number sets contain the number in its database. The searched batch records correspond to all the batches of end-products contributed by the intermediate batch. *T* then publishes the batch indexes of these batch records. By comparing the published

batch indexes with the tag carried recall messages, $\{C_i\}$ could identify the corresponding batches of end-products.

B. Security requirement analysis

Production privacy: Recall in section II.C, we discuss a straightforward approach to generate recall messages, which leaks two types of production information: (1) how many intermediate batches are used to produce an end-product (by counting the number of a batch number set), (2) if two end-products are contributed by same intermediate batch(es) (by comparing two batch number sets). In COLLECTOR, instead, *T* generates recall messages without the knowledge of batch number sets. In specific, considering the generation of a recall message: (*idx*, $<CT>_{\mathsf{sig}}$). *T* encrypts a witness message (m_t, r_l), which is independent of batch number sets, to the symmetric-ciphertext *CT* by using a randomly generated secret key *K*. *T* then signs *CT* by using its signing key. Finally, *T* appends a self-selected batch index *idx* to $<CT>_{\mathsf{sig}}$. During the above generation process, *T* does not use batch number sets as input. A recall message thus does not reveal any information which can be inferred from batch number sets, such as the above two types of production information.

Recall authentication: In the recall phase of COLLECTOR, the secret keys of identification message only enable a C_i to decrypt the recall messages of problematic end-products to get witness messages. Each witness message convinces *T* that an end-product is truly problematic. As a result, to return a good end-product to pass *T*'s authentication, a malicious C_i should be able to get a witness message for the product. In COLLECTOR, each witness message (m_t, r_l) contains a unique random number r_l and is encrypted (through symmetric encryption) into a recall message. Due to the confidentiality of symmetric encryption scheme, the C_i cannot decrypt the recall message of the product without the secret key. Also due to the randomness of r_l, the C_i cannot guess the encrypted witness message or create a witness message by just coping the recovered witness message of a problematic end-product.

Recall non-repudiation: In the recall phase of COLLECTOR, If a C_i identifies an end-product as problematic, the C_i can use its recall message and the identification message to create a recall-evidence for the end-product. Later, if T refuses to recall this end-product, the C_i can reveal the recall-evidence to an authority for judgement. With the recall-evidence ($<CT>_{\mathsf{sig}}$, $<K>_{\mathsf{sig}}$, $<m_t>_{\mathsf{sig}}$), the authority convinces that the evidence truly corresponds to a problematic end-product through following steps:

- verify if $<CT>_{\mathsf{sig}}$ is correctly signed by *T*
- verify if $<K>_{\mathsf{sig}}$ and $<m_t>_{\mathsf{sig}}$ are correctly signed by *T*
- decrypt *CT* to (m'_t, r'_l) by using *K* and decide if $m'_t = m_t$

Let us analyze what the authority can convince from the above three steps. In the first step, the authority concludes that the ciphertext *CT* is generated by *T* based on the unforgeability of signature scheme. This convinces the authority that the C_i truly buys an end-product from T. In the second step, the authority concludes that the recall token m_t and the secret key K are generated by *T* based on the unforgeability of signature scheme. This convinces the authority that a recall event truly happens. Finally, in the third step, the authority concludes that the witness message (m_t, r_l) can be decrypted from *CT* by using *K*. In COLLECTOR, the successful decryption indicates that an end-product is identified as problematic. However, the C_i can easily fake a tuple (m'_t, *CT*′, *K*′) satisfying decryption relation. As the authority has verified that the tuple (m_t, *CT*, *K*) is generated by *T*, this possibility is precluded. As a result, the authority convinces that the C_i truly buys an end-product from *T* which is identified as problematic in the recall event.

TABLE I
COMPUTATION OVERHEAD

	T	$\{C_i\}$
Production phase	**kg** batch-level **enc+sign** item-level	\
Distribution phase	\	**veri** item-level
Tracking phase	\	\
Recall phase	**sign** batch-level	good: \ prob: **dec**+2**veri** item-level

C. Efficiency requirement analysis

Key management: We analyze the key management overhead in the four phases of COLLECTOR. In the first two phases, namely production phase and distribution phase, the end manufacturer T needs to securely store a signing key and a secret key *K* of a symmetric encryption scheme for each of its batches (contained in the batch records). On the other hand, the customer set $\{C_i\}$ do not need to securely store any keys at all. When a recall event happens, the subsequent two phases, namely tracking phase and recall phase, are conducted. In these two phases, both T and $\{C_i\}$ do not need to generate any new keys.

Computation: We analyze the computation overhead of COLLECTOR in two aspects: required cryptographic algorithms and computation delay raised by these algorithms.

COLLECTOR builds on two prevalent cryptographic primitives: symmetric encryption scheme and signature scheme. Therefore, five types of cryptographic algorithms: key generation-encryption-decryption algorithms of the symmetric encryption scheme and signing-verification algorithms of the signature scheme are required in the four phases of COLLECTOR. We use **kg**, **enc**, **dec**, **sign**, **veri** to denote the computation delay of the five types of algorithms respectively.

In the production phase, the end manufacturer *T* needs to choose a secret key for each of its batches of end-products, incurring **kg** delay in batch-level. *T* also needs to generate a recall message for each of its end-products, incurring **enc+sign** delay in item-level. On the other hand, the customer set $\{C_i\}$ do not need to execute any cryptographic algorithms, incurring no computation delay.

In the distribution phase, *T* directly distributes end-products to $\{C_i\}$, and thus does not need to execute any cryptographic algorithms. On the other hand, the customer set $\{C_i\}$ need to

verify if the recall messages of their received end-products are correctly signed by *T*, incurring **veri** delay in item-level.

In the tracking phase, a C_i just returns a problematic end-product to T, incurring no computation delay. On the other hand, *T* reads the recall message of the reported problematic end-product from the attached tag and uses the batch index of the recall message to search a proper batch record in its database, also incurring no computation delay.

In the recall phase, *T* needs to sign the secret keys of its batch records whose batch number sets contain the problematic batch number to generate identification message, incurring **sign** delay in batch-level. On the other hand, each $C_i \in \{C_i\}$ needs to identify if it has problematic end-products. We classify the end-products of a C_i as good and problematic. For the good end-products, the C_i identifies them through batch index comparison, incurring no computation delay. For the problematic end-products, the C_i identifies them through batch index comparison and creates a recall-evidence for each of them, incurring **dec**+2**veri** delay in item-level.

Table I summarizes the computation overhead of COLLECTOR. From the above analysis, it can be seen that COLLECTOR works in a fine-grained manner in the recall phase: $\{C_i\}$ process good end-products with no computation delay. This means that COLLECTOR is especially efficient when a small fraction of end-products are problematic and need to be recalled.

Tag overhead: In COLLECTOR, RFID tags are not required to implement any cryptographic algorithms. Instead, a tag only needs to support basic read and write operations. We then analyze storage overhead of RFID tags in the four phases of COLLECTOR. The production phase involves the production of end-products and tags are not used. In the distribution phase, tags are attached to end-products in item level with each tag storing a recall message. A recall message consists of a batch index *idx*, a symmetric-ciphertext *CT* and a signature signed on *CT*. In the tracking phase, no new messages are added into the tags. In the recall phase, when an end-product is identified as problematic, its attached tag is re-written with a witness message, which consists of a recall token m_t and a random number r_l. In conclusion, tag storage overhead is constant in the four phases of COLLECTOR.

To suit the storage constraint (512 bits) of commodity C1G2 RFID tags, we can adopt compact 128-bit witness messages and 512-bit recall messages. A 128-bit witness message consists of a 64-bit recall token and a 64-bit random number. A 512-bit recall message consists of a 32-bit batch index, a 128-bit ciphertext and a 320-bit signature. We use 32-bit batch index to support up to 2^{32} batches of end-products. We select AES as the symmetric encryption scheme to encrypt 128-bit witness messages, which generates 128-bit ciphertexts. Finally, we select DSA as the signature scheme to generate 320-bit signatures (in 80-bit security).

VI. Implementation on commodity C1G2 RFID tags

We implement COLLECTOR with commodity C1G2 RFID tags. Since recall messages are longer than witness messages, we only focus on recall messages.

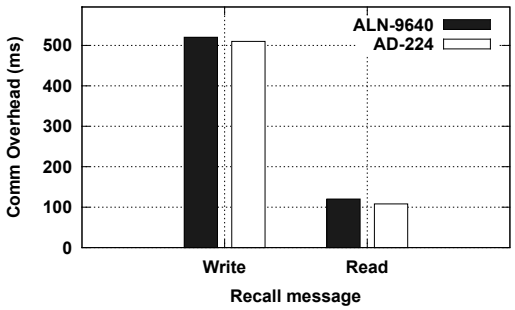


Fig. 4. Communication time of recall message transfer.

The C1G2 standard specifies a set of communication primitives between RFID reader and RFID tags. An RFID reader can write/read recall message to/from an RFID tag following C1G2 standard. In the message transfer procedure, the reader may write recall messages into RFID tags using the Write command which allows the reader to write a 16-bit data block per operation. To transfer more data, the reader needs to write multiple data blocks. If the recall message is successfully saved into the tag memory, the tag will notify the reader by sending an acknowledgement. The Read command on the other hand supports bulk data collection. Current commodity RFID reader supports up to 512-bit data collection per Read operation.

We use the Alien ALR 9900+ commodity RFID reader to write and read recall messages to commodity passive RFID tags. We adopt the default setting of RFID reader with transmission power of 30dBm. The reader is equipped with one ALR-8696-C directional antenna. The data transfer program is developed based on the Alien RFID reader SDK codes. Our implementation only requires the C1G2 routine operations such as writing and reading data from tags. Thus, we believe COLLECTOR can also be implemented on other commodity RFID systems.

Current commodity RFID tags have different sizes of non-volatile memory which can be used to store recall message. In particular, there are 4 different types of memory banks in commodity RFID tags, i.e., Reserved, EPC, TID, and User. In our implementation, we store recall message in the user memory bank. We test with two different types of passive tags from two different manufacturers – ALN-9640 and AD-224 tags both with 512-bit user memory. As COLLECTOR uses compact 512-bit recall message, the user memory can comfortably accommodate recall message. COLLECTOR does not require any modifications to the commodity passive RFID tags or implement additional cryptographic functionality on tags.

Next, we focus on the communication overhead between the reader and the tags in message transfer. As COLLECTOR uses the C1G2 Write/Read primitives, the performance is largely dictated by the performance of commodity systems. Fig. 4 shows the communication overhead involved in the 512-bit

recall message transfer. According to the experiment results, it requires more time to write messages into tags because as mentioned the Write command only allows the reader to write a 16-bit data block per Write operation. To transfer 512-bit data, the reader needs to first divide the recall message into several blocks and transfer them separately which takes longer time. In comparison, the Read operation takes less time since it only requires one Read operation to collect the whole 512-bit recall message.

VII. RELATED WORK

Tag authentication schemes [3], [4], [5], [6], [7], [8], [9], [10] are proposed to provide authentication for RFID system. Weis et al. [3] propose a provable secure authentication protocol, Hash Lock, to authenticate tags with privacy guarantee. The drawback of Hash Lock is that its key search complexity is proportional to the number of tags in the system. To solve this problem, Tree based schemes [4]–[6] have been proposed, which improve the key search efficiency from linear complexity to logarithmic complexity. Yao et al. [7] investigate the common key effect which is a privacy concern in RFID system and present a random walk based authentication protocol to enhance the privacy and security. Recently, several works [9], [10] investigate the authentication of batches of tags with high efficiency performance. For instance, Yang et al. [10] propose a batch authentication protocol to detect fake tags in tag batches. Their technique relies on the distribution of tag replies to fast detect fake tags with probabilistic guarantee.

Secure schemes in RFID-enabled supply chains [11], [12], [13] investigate the secure issues when a large quantity of tags flow through multiple parties in a supply chain. Li et al. [11] identify the security requirements of a general RFID-enabled supply chain and propose an authentication protocol to satisfy these requirements. Juels et al. [12] consider the key distribution problem on an unidirectional channel through an RFID-enabled supply chain and present two unidirectional key distribution schemes with different security guarantees. Blass et al. [13] propose a secure framework, tracker, to solve the path authentication problem in RFID-enabled supply chain. The goal of tracker is to authenticate whether a tag has flowed through a valid processing path.

The closest work with ours is [14]. In [14], the authors propose a solution to protect industrial privacy in RFID-enabled batch recall. Comparing with our work, their solution considers a totally different threat model where intermediate manufacturers do not trust each other in the production process. Also, they do not consider several requirements such as recall authentication and recall non-repudiation which are desired by us.

Finally, to make our protocol design manageable, we do not consider attacks from network, such as internet worm attack [15], [16]. An open question is if and how network attacks may affect the functionality of COLLECTOR. We also ignore the collision problem of RFID system as it has been investigated by many prior work. For instance, Yang et al. [17] proposed an anti-collision protocol to improve the identification efficiency for densely deployed RFID systems.

VIII. CONCLUSION

RFID-enabled batch recall allows us to identify problematic products and enables efficient recall operations. Direct batch recall solutions however usually save the sensitive production information in RFID tags which necessitates a careful design to provide security. In this paper, we thoroughly analyze the requirements of RFID-enabled batch recall including function requirements, security requirements and efficiency requirements. We then propose a secure RFID-enabled batch recall protocol, called COLLECTOR to fulfill these requirements. We conduct comprehensive analysis and prove the correctness and efficiency of the proposed protocol.

IX. ACKNOWLEDGEMENT

We acknowledge the support from Singapore MOE AcRF Tier 1 grant MOE2013-T1-002-005, NTU Nanyang Assistant Professorship (NAP) grant M4080738.020 and National Natural Science Foundation of China (NSFC) (No. 60933003).

REFERENCES

[1] [Online]. Available: http://www.notebookreview.com/default.asp?newsID=3212&article=Lenovo+ThinkPad+Battery+Recall
[2] [Online]. Available: http://en.wikipedia.org/wiki/Toyota_vehicle_recalls
[3] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," in *SPC*, 2003.
[4] L. Lu, J. Han, R. Xiao, and Y. Liu, "Action: Breaking the privacy barrier for rfid systems," in *INFOCOM*. IEEE, 2009.
[5] L. Lu, J. Han, L. Hu, Y. Liu, and L. M. Ni, "Dynamic key-updating: Privacy-preserving authentication for rfid systems," in *PerCom*. IEEE, 2007.
[6] T. Dimitriou, "A secure and efficient rfid protocol that could make big brother (partially) obsolete," in *PerCom*. IEEE, 2006.
[7] Q. Yao, Y. Qi, J. Han, X. Li, and Y. Liu, "Randomizing rfid private authentication," in *PerCom*. IEEE, 2009.
[8] T. Dimitriou, "A secure and efficient rfid protocol that could make big brother (partially) obsolete," in *PerCom*. IEEE, 2006.
[9] W. Gong, K. Liu, X. Miao, Q. Ma, Z. Yang, and Y. Liu, "Informative counting: fine-grained batch authentication for large-scale rfid systems," in *MobiHoc*. ACM, 2013.
[10] L. Yang, J. Han, Y. Qi, and Y. Liu, "Identification-free batch authentication for rfid tags," in *ICNP*. IEEE, 2010.
[11] Y. Li and X. Ding, "Protecting rfid communications in supply chains," in *ASIACCS*. ACM, 2007.
[12] A. Juels, R. Pappu, and B. Parno, "Unidirectional key distribution across time and space with applications to rfid security," in *USENIX Security*, 2008.
[13] E. Blass, K. Elkhiyaoui, and R. Molva, "Tracker: Security and privacy for rfid-based supply chains," in *NDSS*, 2011.
[14] L. W. F. Chaves and F. Kerschbaum, "Industrial privacy in rfid-based batch recalls," in *Enterprise Distributed Object Computing Conference Workshops*, 2008.
[15] Y. Yao, X. Xie, H. Guo, G. Yu, F. Gao, and X. Tong, "Hopf bifurcation in an internet worm propagation model with time delay in quarantine," *Elsevier Journal of Mathematical and Computer Modelling*, vol. 57, no. 2635-2646, June 2013.
[16] Y. Yao, N. Zhang, W. Xiang, G. Yu, and F. Gao, "Modeling and analysis of bifurcation in a delayed worm propagation model," *Journal of Applied Mathematics*, vol. 2013, 2013.
[17] L. Yang, Y. Qi, J. Han, W. Cheng, and Y. Liu, "Shelving interference and joint identification in large-scale rfid systems," *IEEE Transactions on Parallel and Distributed Systems*, 21 Nov. 2013.