Routing Mechanism for Secure, Distributed Discovery Services for Global Auto-ID Networks

Jons-Tobias Wamhoff[1,2], Eberhard Grummt[1,2] and Ralf Ackermann[1]

[1] SAP Research CEC Dresden
{jons-tobias.wamhoff, eberhard.oliver.grummt
ralf.ackermann}@sap.com
[2] Dresden University of Technology

Abstract. Enterprises capture immense amounts of data in Auto-ID repositories as items travel along the supply chain. For collaboration among partners individual repositories need to be discoverable when they are not explicitly linked. At the same time confidential information including supply relationships needs to be protected. The task of identifying repositories by keeping the secrets of a supply chain is done by a Discovery Service. The discovery information can be spread over multiple nodes. The routing mechanism ensures that queries sent to any node within the Discovery Service hierarchy will be forwarded to all responsible nodes. By anonymizing the request and response propagation, the repositories and Discovery Service data nodes remain hidden as long as the request did not produce an authorized result set.

1 Introduction

Auto-ID networks target on allowing enterprises an integrated supply chain management by capturing data about items automatically (e.g. with RFID technology) and storing it in back end systems for further processing or analysis. This closes the gap between the digital and the physical world. As items travel along the supply chain they leave a trace in the form of events in local repositories of organizations that processed a business step. But as long as the observed data is kept local and traces are not linked explicitly it is not possible to engineer applications for Track and Trace [1]. Therefore, partners that belong to a supply chain but not to the same enterprise need the possibility to discover and retrieve item information that is distributed among several repositories. This task is fulfilled by an additional Discovery Service that locates all repositories that contain information for an item of interest. A typical application of Track and Trace is a recall of a product where it is necessary to locate all resellers or even customers that own the product.

This paper proposes a routing mechanism for a Discovery Service that allows to protect a supply chain by keeping repository and Discovery Service anonymous as long as the requester is not entitled for access. Because collaboration requires commitment to open standards this paper is based on standards of the *EPCglobal Architecture Framework* [9]. It defines a naming scheme for *Electronic Product Codes (EPC)*, and presupposes the existence of data repositories according to the *EPC Information Service (EPCIS)* specification [5]. Queries can be sent by EPCIS Accessing Applications

(EPCIS AA) which can be from within an enterprise or from a external partner. For locating the *EPC Manager* that originally assigned an EPC, the *EPC Object Naming Service (ONS)* [4] can be used. ONS does hardly addresses any privacy or security aspects and is not sufficient for item discovery [6]. A standard for *EPC Discovery Service (EPCDS)* is not available yet.

The rest of the paper is structured as follows. Section 2 introduces hierarchical architecture topologies and the new approach towards secure routing of Discovery Service requests. Sections 3 and 4 discuss perspectives for this new approach and show related work.

2 Hierarchical Routing

Organizing Naming Services and Directory Services in a hierarchical structure is widely used and well examined, e.g. in the context of DNS. It has the advantages to be adaptive to higher storage or timing demands by adding new branches to the tree. To simplify administration of EPCDS the origin of information to resolve must reside in leafs of the tree only (lowest level of hierarchy).

The EPCDS tree itself is used for routing requests to the responsible EPCDS nodes. Privacy requires the anonymity of both the requester of information about an EPC as well as the responder that hosts and offers this information. The aim is that the requester should not be identified during the routing process and the EPCDSs have to be hidden in the case they do not allow access for the requester. To grant or deny access it is necessary to get the ID of the requester in the EPCDS leaf.

A request consists of the ID of the requester, the EPC and optional predicates such as a time frame or business step. Routing is entirely based on EPCs. Registrations and queries that contain EPC ranges are split in multiple requests that are processed sequentially.

To allow the superior approach of a Directory Service over a Daisy-chain, which means to refuse explicit linking of predecessors and successors in a supply chain, the EPCIS repositories need to self-register with the EPCDS on the first occurence of an EPC. All registrations for a single EPC can either be stored in one EPCDS or they are distributed over a federation of EPCDS, e.g. for more locality or fault-tolerance. The location of the EPCDS information depends on the agreement of supply chain partners. If they host their own EPCDS, it must be guaranteed that internal or external partners are granted write access to make their repository information publicly available.

2.1 Encrypted Routing Tree

To gain a better protection of supply-chain information together with an increase in anonymity of the Discovery Service from the clients we propose an advanced routing architecture. The Discovery Service will do query and result routing internally and consists of two different kinds of nodes: routing nodes and the actual EPCDS data leaf nodes (DSd). EPCDS nodes must not provide routing functionality under their identity to remain anonymous. Routing nodes have two interfaces: DS entry (DSe) for accepting client requests and DS route (DSr) for accepting routing requests. In detail processing of a request works like this (Figure 1):

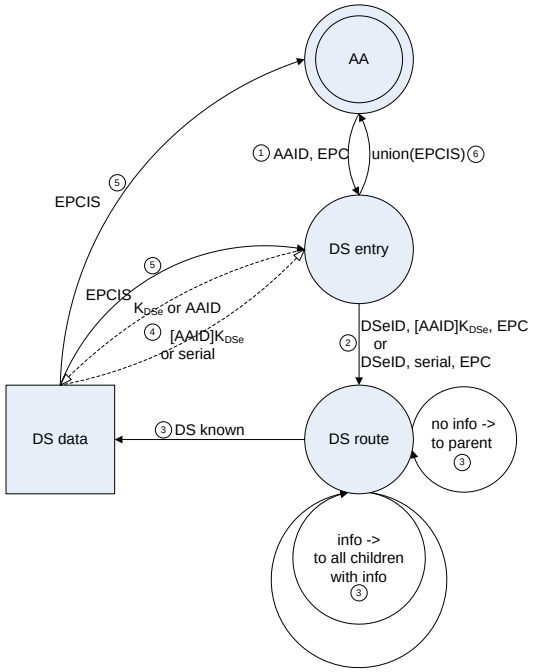

Fig. 1. Schema of encrypted routing tree.

1. EPCIS AA (AA) contacts any known DSe node with its ID and the EPC
2. DSe encrypts AA ID with its own key to hide the identity of AA (key of target data EPCDS leaf can not be used because it is unknown). Contacted EPCDS becomes new originator of request, which contains the plain EPC and DSe ID along with the encrypted AA ID. The new request is forwarded to the own or any other DSr node.
3. Request is routed according to EPC: if DSr has no routing information the request is forwarded to the parent DSr, if DSr has routing information about the EPC it forwards the request to all children that are connected with the EPC in the routing table or, if DSr is aware of a DSd, the request will be forwarded to that location. This will be repeated at every DSr until the request reaches a DSd.
4. DSd uses DSe ID to get its key (symmetric encryption) by contacting the node directly, authentication is done with certificates. To avoid that an attacker can obtain the key illegally, the contacted EPCDS should cache the EPCs that are in progress and the key should not be returned more than once. Black and whitelists can be used to ban fraudulent nodes.
5. DSd uses AA ID for authorization and has now two possibilities to return the result, either by sending it back to DSe or directly to AA (then skip the remaining step).
6. DSd returns union of all received results to DSe and DSe delivers result to AA.

All message channels are link encrypted. The DSe key is valid for one transaction only or if nodes have frequent contact and trust each other they exchange a key that

is valid on a longer term. Alternatively to encryption, DSe can use serial numbers that are kept in a cache with the corresponding AA ID and DSd would request the AA ID instead of the DSe key. While requests can contain more predicates than EPC (e.g. business step) that will be forwarded to DSd, routing is based on EPC only.

Discovery information about one EPC can be spread over multiple DSd, so the DSr nodes will split the request according to their routing tables. The merging is to be done at AA if the results are directly returned to the client or at DSe. If the results are routed over DSe, it is possible to allow asynchronous communication. If no information is available or AA is not authorized to get information about the EPC, the result will be a timeout and the DSd remains anonymous. If the result is directly returned to AA, DSd is exposed and loses its anonymity towards an authorized client. The advantage is that AA can cache the DSd for further requests, again the timeout makes it indistinguishable if the new request was not authorized or not successful. Also in case of successful requests, DSd can include a certificate that proves authenticity.

2.2 Onion Routing Tree

Onion routing is known from Tor [3] and other proxies for anonymizing Internet connections. In principle, the topology of the onion routing tree is similar to the encrypted routing tree. The difference is that in every routing step the request will be encoded again using the key of the current DSr, adding one layer to the onion. When the request reached DSd, the node needs to peel the onion skin layer-wise by stepwisely collecting all keys of DSr nodes that processed the request. The key of the outermost encrypted skin is not encrypted and can be used to contact a Key Distribution Center (KDC) for the key of that node. The key is used to decrypt the next skin layer which contains the next DSr ID or the DSe ID that finally leads to the AA ID that is required for authorization. All DSr IDs and DSe ID together with their keys are cached for result routing.

Since DSd will never be exposed to AA it can return empty results even if AA is not authorized to get any information. The result is encoded in the opposite order it was peeled and transmitted to the first DSr which removes its skin by decrypting the result message and forwards it to the next visible DSr. If a request was split in a DSr it waits for all responses from children to whom requests have been forwarded. Failed request (e.g. expired timeout) can be resubmitted, potentially with replicated nodes. When the result finally reaches DSe, the last layer is peeled. DSe does a union on all received results and delivers it to AA.

3 Related Work

Naming Services such as DNS resolve a name to an address or id. Thereby no access control is done. Discovery Services are used to locate a service in a network that is specified by properties. In contrast to known Discovery Services, a EPCDS has to include access control and privacy protection of EPCIS AA and EPCIS.
Agrawal et al. [1] define traceability as the ability to track the current and all previously recorded stated of an object. The approach presented requires supply chains to be explicitly defined by additional attributes. The main drawbacks of explicit chaining are

that if the node of a partner fails then the supply chain cannot be reconstructed and that every participant must employ a compliant system.
Beier et al. [2] describe the role of an EPCDS in a EPC network and the requirements for privacy and security. EPCDS is seen as an independent unit at which each EPCIS repository has to register with the EPC when they observe an item for the very first time. It is also mentioned that EPCDS should allow row-level data access control. The presented requirements match those for this work but no details of the prototype architecture are given.
Huang et al. [7] present a distributed architecture that makes use of peer-to-peer technology and distributed hash tables to locate information about explicitly defined supply chains.
Kürschner et al. [8] give an overview about requirements towards an EPCDS and describe the directory look-up design currently under consideration at EPCglobal.

4 Summary and Future Work

In this paper we presented hierarchical architecture approaches that allow to keep discovery information related to single EPCs to be spread over multiple EPCDS nodes. EPCDS and EPCIS AA remain anonymous from each other as long as a request produces no authorized result set because EPCIS AA and EPCDS do not interact directly with each other. Instead, requests are routed through a tree of nodes that re-encode the request and change its originator. For authorization needs the EPCIS AA has to give up its anonymity when the request reaches a responsible EPCDS data node but the EPCDS can stay anonymous at any time. On an empty response it is indistinguishable for EPCIS AA whether access was denied or no information for the EPC was available.

Future work includes how routing tables in DSr nodes are organized and how they can be compressed for more efficiency. It also should be investigated if the hierarchical approach with routing tables can be replaced by a peer-to-peer approach with distributed hash tables as the routing mechanism. It has to be ensured that a request does a sufficient amount of hops to keep its anonymity. For both approaches a prototype has to be implemented for performance measurements.

References

1. Rakesh Agrawal, Alvon Cheung, Karin Kailing, and Stefan Schönauer. Towards Traceability across Sovereign, Distributed RFID Databases. In *IDEAS '06: Proceedings of the 10th International Database Engineering and Applications Symposium*, pages 174–184, Washington, DC, USA, 2006. IEEE Computer Society.
2. Steve Beier, Tyrone Grandison, Karin Kailing, and Ralf Rantzau. Discovery Services – Enabling RFID Traceability in EPCglobal Networks. In *Proc. of the 13th International Conference on Management of Data (COMAD)*, Delhi, India, December 2006.
3. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.
4. EPCglobal Inc. Object Naming Service (ONS) Version 1.0 Specification, October 2005.
5. EPCglobal Inc. EPC Information Services (EPCIS) Version 1.0 Specification, April 2007.

6. Benjamin Fabian, Oliver Günther, and Sarah Spiekermann. Security Analysis of the Object Name Service for RFID. In *International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU'05*, Santorini Island, Greece, July 2005. IEEE, IEEE Computer Society Press.
7. Dijiang Huang, Mayank Verma, Archana Ramachandran, and Zhibin Zhou. A Distributed ePedigree Architecture. In *FTDCS '07: Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 220–230, Washington, DC, USA, 2007. IEEE Computer Society.
8. Chris Kürschner, Cosmin Condea, Oliver Kasten, and Frdric Thiesse. Discovery Service Design in the EPCglobal Network: Towards Full Supply Chain Visibility. In *The Internet of Things*, volume 4952, pages 19–34. Springer Berlin, Heidelberg, March 2008.
9. Ken Traub, Greg Allgair, Henri Barthel, Leo Burstein, John Garrett, Bernie Hogan, Bryan Rodrigues, Sanjay Sarma, Johannes Schmidt, Chuck Schramek, Roger Stewart, and KK Suen. The EPCglobal Architecture Framework, July 2005.