

# Practical Access Control Management for Outsourced EPC-Related Data in RFID-Enabled Supply Chain \*

Sergei Evdokimov and Oliver Günther  
Humboldt Universität zu Berlin  
Spandauersrt. 1, 10178, Berlin, Germany  
{evdokim, guenther}@wiwi.hu-berlin.de

## Abstract

*To reduce deployment and maintenance costs companies often decide to outsource parts of RFID infrastructure to an external application service provider. In scenarios where outsourced data can be accessed by several parties, the managing of discretionary access control to it becomes a very important issue.*

*In this paper we present an approach that uses selective encryption for managing discretionary read/write access to the data that describe tagged objects and are submitted by a number of parties to a central datastore. Our approach does not reveal relationships between the parties and allows efficient dynamic assignment of read permissions. We proceed with presenting experimental performance results and based on them discuss the applicability of our approach to practical scenarios.*

## 1 Introduction

One of the most promising applications of RFID technology is automated identification of objects. Use of RFID in a supply chain allows to optimize logistics and inventory management processes, increase productivity and reduce out-of-stock losses. As a result, the number of companies that introduce RFID technology is constantly growing and with every year more and more items tagged with RFID tags are moving along supply chains.

To standardize the use of RFID and make the RFID tags readable worldwide non-profit consortium EPCglobal developed the Electronic Product Code (EPC) standard that defines a family of coding schemes that can be employed for identification of objects tagged with a Class 1 Generation 2 RFID tags [7]. On addition, the EPC number might serve as a reference to EPC-related data that provide details about

the tagged item (manufacturing date, delivery date, price etc.). A datastore where the EPC-related data are stored is usually referred to as *EPC repository*. The ability to globally access or provide EPC-related data is achieved by the means of Object Naming Service (ONS) that resolves an EPC number into an address of the corresponding EPC Information Service (EPCIS) that is responsible for retrieval EPC-related data from the EPC repository.

However, in spite of all the virtues such global infrastructure has to offer, the companies that already use or plan to introduce RFID, though considering EPCglobal standards, do not hurry to make EPC-related product data globally available. Instead, all EPC-related data are stored in a central datastore that is accessible exclusively to supply chain partners. For example, Wal-mart, a major retail chain that was helping to develop ONS and pioneered in deployment of RFID in its stores is not going to make ONS a part of its RFID infrastructure. Instead, EPC-related data are stored on Wal-mart's servers and accessed by its suppliers via EDI. U.S. Ministry of Defence, which also contributed a lot in development of RFID standards, is planning to follow the same approach.

There are several reasons that explain why companies choose to deploy RFID infrastructure without supporting ONS. The most important is, probably, an asymmetry between expenses that are required to deploy an ONS-compliant infrastructure and its benefits. According to a report presented by A.T. Keartney RFID brings great benefits for retailers, whereas manufacturers get little return and carry significant maintenance burden [2]. As a result, manufacturers often see no reason in hosting EPC-repositories and obtaining EPCglobal membership.

Also, the centralized approach provides a simple and straightforward way to share and collaboratively use the EPC-related data. In cases when data related to the same EPC number resides on several repositories EPCglobal proposes to use EPC Discovery Services for locating EPCISs that could retrieve these data. However, at the time of this writing, EPC Discovery Services are in "To Be De-

---

\*Published In ICEBE'07: Proceedings of IEEE International Conference on e-Business Engineering 2006

fined” status and only preliminary information describing their role in EPCglobal network was released.

On the other hand the centralized architecture neither requires special services for locating EPC repositories of the partners nor it burdens the manufacturers with the necessity to host the repositories - the EPC repository may be hosted by any of the partners or outsourced to an application service provider (ASP) (in case the repository is hosted by a supply chain participant, hereinafter we refer to such participant as to ASP as well). Also, having all the data in one place makes it straightforward to share it between the supply chain participants.

However, before the centralized EPC repository is deployed the supply chain participants have to address the following privacy issue: Certain portions of the EPC-related data may be considered as sensitive and its owners might want to prevent some of the partners from seeing them. Since the data are stored in an outsourced repository on which the supply chain participants have no direct control, there arises a need to manage data access rights. In this paper we address this problem and present a practical solution that allows the data owners to selectively restrict read access to the outsourced EPC-related data and prevent it from being unauthorizedly modified.

## 2 Privacy issues

In certain cases the supply chain participants will want to selectively restricts access to the EPC-related data they submit to the central EPC repository. As an example, consider a manufacturer that needs to keep track of a factory that produced a product and, thus, stores id of the factory in the EPC repository. However, in order to prevent a discrimination of the products based on their origin, the manufacturer considers this information as private. Or suppose that the manufacturer is agreed to share this information with a wholesaler but a retailer and logistic companies that also provide services to competitors should have no access to it.

It is natural to assume that the EPC repository will be implemented as a database. Usually the problem of database access control is formulated as an ability of a database administrator to manage access privileges of database users [12]. In most of the *de facto* industry standard database management systems the discretionary access control is implemented by means of privileges, roles, views, stored procedures and virtual private databases [3]. These approaches suppose that the database administrator has full access to the data. When the EPC repository is outsourced the supply chain participants should be aware that their product information will be stored on servers on which they have no direct control. In case the ASP is not trusted, the discretionary data access should be regulated by mechanisms that are not dependent on ASP’s honesty.

In the next chapter we review existing works and discuss their applicability to the RFID-nabled supply chain.

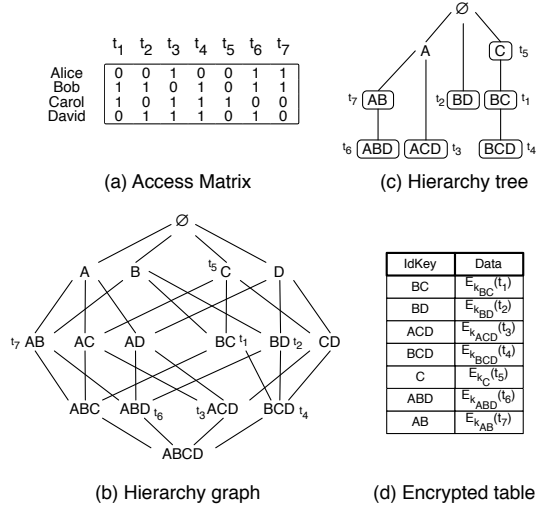
## 3 Related work

The Database-as-a-Service (DAS) paradigm where a database is outsourced to the ASP gave rise to several specific privacy problems. So, it may happen that the outsourced data is sensitive and its owner may be willing to restrict the ASP from seeing it. The straightforward solution is to rely on legal mechanisms and enforce the privacy of the data via a contract. However, there exist scenarios where such mechanisms may be not sufficient. In case of a takeover the database may be transferred to a new owner as an asset of the purchased company. Or the ASP may purposely brake the privacy agreement since such intrusion is virtually undetectable. Generally, the works proposing solutions to this problem can be classified in two general approaches: The weak encryption of the database that does not conceal all information about the data and in that way allows processing of some queries [9] and encryption of the database with a secure encryption scheme that allows to perform search on the encrypted data [4, 8, 13]. However, no solution that could provide both sufficient levels of security and functionality has been proposed so far: The former approach allows to execute a broad spectrum of queries but provides no security guarantees, the latter approach, on the contrary, guarantees a high level of security but gives a very limited functionality.

Another issue that is often mentioned when referring to the DAS paradigm is the necessity to provide a discretionary access to the outsourced data. And again, if the ASP is trusted, the straightforward approach is to rely on already mentioned security mechanisms natively supported by the DBMS and let the database administrator manage the access rights. Otherwise one should use a solution that does not depend on the honesty of the ASP.

A number of solutions, at least partially solving this problem, were presented over the last few decades. Most of them deal with access control in a tree-like hierarchy [1, 10, 11, 14] that is a special case of a general access control problem. The underlying idea of these solutions is to selectively encrypt tuples with keys that are derivable from each other according to the hierarchy: Keys corresponding to the lower hierarchy levels can be derived from keys assigned to a higher level but doing it in the opposite direction is computationally infeasible. However, these solutions cannot be applied in our case since in a supply chain access permissions of partners not necessary constitute a hierarchical structure.

In [5] Damiani et al. consider a very similar problem and present a solution that allows to implement access control in a multi-user database with arbitrary defined tuple-wise



**Figure 1. Damiani et al. Algorithm**

access permissions. Using an example we briefly outline their solution.

The authors consider a table in a multi-user environment where each user is assigned a set of privileges that give her access to a subset of tuples. In our example we consider a table consisting of seven tuples  $t_1, \dots, t_7$  with permissions defined by an access matrix (Figure 1(a)) where  $ij$ th element is equal to 1 if  $i$ th user can access tuple  $t_j$  and to 0 otherwise. The set of access privileges is represented as a directed acyclic graph (Figure 1(b)) that by means of a heuristic algorithm is transformed into a hierarchy tree (Figure 1(c)). Each node of the tree is assigned a key that is produced as an output of a pseudo-random function with a key that is a name of the parent node:  $k_{n_i} = f_{\text{name}(n_i)}(k_{n_j})$  where node  $n_j$  is a parent of node  $n_i$  (e.g.  $k_{AB} = f_{AB}(k_A)$ ). For the root node the key is randomly generated. The keys are used to encrypt the corresponding tuples (Figure 1(d)). By making the keys derivable from each other the algorithm reduces the number of keys a user has to store in order to be able to decrypt tuples to which she has access.

However, there is a number of issues impacting security and performance of this solution. So, as the authors note themselves, changes in access rights, users and objects will often lead to changes in the structure of the hierarchy tree which result in the necessity to redistribute the keys and re-encrypt the data. Appending each tuple with the lists of users that can access it exposes the relationships between the users. Finally, the existence of the root key implies the existence of a party that has full access to the data.

All these issues can matter when talking about access control to the EPC-related data in the supply chain. The

relationships in the supply chain are often short-term. In some cases the participants might want to keep secret the fact that they are sharing some information. Also the problem we are considering does not suppose the existence of a party that is trusted and may access all the data. Therefore there is still a need for a solution that overcomes the aforementioned problems and allows to efficiently and flexibly manage access rights to the outsourced EPC-related data.

## 4 Access control to EPC-related data

In the previous section we have mentioned a number of approaches that solve the problem of discretionary access control to the outsourced data. The primary aim of these works was to allow the data owners with the aid of cryptographic methods to maintain exclusive control over the access to their data while minimizing the amounts of keys the users have to manage. In many respects it was done at the expense of narrowing the original problem to a more specific scenarios where the users are organized in hierarchies, the relationships between the users are an open information and data and access permissions are static. While seeming reasonable in some scenarios, for the supply chain it often might be necessary to handle arbitrary patterns of user relationships, keep these relationships confidential and quickly react to structural changes.

In our work we employ the similar approach: Encrypt the data with different keys and distribute these keys between the users in such a way, that they can only access data to which they are granted appropriate permissions. But unlike the existing solutions our main objective is not to reduce the number of the keys but to be able to handle arbitrary dynamic patterns of the access permissions, make the solution easy to implement and efficient enough to be applicable to existing supply chains.

We assume that the EPC number is not considered as a sensitive information, the EPC number serves as a primary key in a table that stores EPC-related data and is used as a selection criterion in exact select queries.

We start with a simplified version of the algorithm that is applicable to a particular configuration of the supply chain and conclude with an algorithm that is applicable for general scenarios.

### 4.1 Read Access Control in a Simplified Scenario

Suppose that the supply chain has a “linear” structure and each stages (manufacturing, distributing, retailing etc.) consists of a single participant. The table that contains EPC-related data consists of a primary key attribute  $attr_{EPC}$  that stores EPC numbers and a set of attributes  $A = \{attr_1, \dots, attr_n\}$ . Let  $U_1, \dots, U_n$  be sup-

ply chain participants that access the EPC repository. Every supply chain participant  $U_i$  owns an attributes subset  $\mathcal{A}_i = \{attr_{i_1}, \dots, attr_{i_{n_i}}\}$ ,  $\mathcal{A}_i \subset A$  and is responsible for providing values for the owned attributes. Due to the linear structure of the supply chain there is no attributes that are shared between the participants:  $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ .

In such linear case in order to allow the participants to control access to the owned data it is sufficient to introduce an attribute-wise access control mechanism. It is easily implementable by an attribute-wise encryption of every tuple, where each attribute value is encrypted with a key generated by an attribute owner: User  $U_i$  owns attribute  $attr_{i_k}$  and encrypts its values with key  $k_{i_k}$  using a symmetric encryption algorithm. If user  $U_i$  decides to grant user  $U_j$  access to the values of attribute  $attr_{i_k}$ ,  $U_i$  shares key  $k_{i_k}$  with  $U_j$ .

## 4.2 Read Access Control in a General Scenario

Now we extend the algorithm so that it could provide discretionary access control in general scenarios where the supply chain can be modelled as an arbitrary weakly connected graph.

Unlike the scenario where the supply chain has a linear structure the attribute-wise access control is obviously not sufficient here since each stage may now consist of more than one participant. That means that values of the same attribute can be submitted by different parties that may be setting different access permissions on them, thus, making it necessary to implement the tuple-wise access control.

A straightforward way to implement it is to have the attribute values encrypted by the owners, distribute the keys according to the access permissions and let the user know which key should be used for decryption by accompanying each attribute value with an id of its owner.

The approach may require users to store large numbers of keys, but considering a relatively short bit length of the keys used by symmetric encryption schemes<sup>1</sup> it should not be considered as a significant drawback.

Being quite simple and efficient, yet such approach fails to securely protect sensitive attribute values. The identifier of the owner allows to single out values submitted by the same user and launch a statistical attack that can help to guess the encrypted values.

However, if the owner identifiers are not stored together with the encrypted attribute values the users are not able to single out the appropriate key and, thus, should perform a bruteforce search among the keys that could be used for encrypting this value (*candidate keys*). The search is performed by sequentially decrypting the encrypted value with

<sup>1</sup>According to NIST recommendations AES encryption algorithm with 128 bits key is sufficient for encrypting data with security lifetime beyond year 2030.

each candidate key until one of the keys produces a valid plaintext. To make the valid plaintext easily recognizable an attribute value  $x$  can be prepended with the corresponding EPC number  $epc$  that is separately encrypted with the same key ( $E_k(epc)|E_k(x)$ ) or, in case a block cipher in one of the chaining modes is used, padded up to be a multiple of the cipher's block size and encrypted together with  $x$  ( $E_k(epc_{padded}|x)$ ). Now to find a valid key it is sufficient to be decrypting only a part of the ciphertext which contains the EPC number. Further in the paper we assume that the latter approach is used.

To summarize, consider users  $U_{i_1}, \dots, U_{i_l}$  that are a subset of all users that can submit values of attribute  $attr_j$  and  $k_{i_1j}, \dots, k_{i_lj}$  are the keys with which these users encrypt the values they submit: Key  $k_{ij}$  is generated by user  $U_i$  and attribute value  $x$  submitted by  $U_i$  is stored as  $c = E_{k_{ij}}(epc_{padded}|x)$ . Suppose user  $U$  has read access to values of attribute  $attr_j$  submitted by  $U_{i_1}, \dots, U_{i_l}$  and, therefore, knows the corresponding keys. When  $U$  receives from the ASP tuple  $\langle EPC : epc, attr_1 : c_1, \dots, attr_n : c_n \rangle$ , in order to read the value of attribute  $attr_j$  she tries to decrypt  $c_j$  by sequentially applying the decryption procedure with keys  $k_{i_1j}, \dots, k_{i_lj}$  to the part of the encrypted attribute value that contains the padded EPC number. She stops either when there is key that produces the valid  $epc$  or after the last key was tried. The former means that  $U$  has read access to this value and the found key can be used to decrypt the rest of the ciphertext, the latter means that the access was denied.

The necessity to look through the set of candidate keys inevitably increases the time needed to process a tuple. We postpone a detailed discussion on the performance and applicability of our approach to the next section and only mention that it is comparable to the performance of Damiani et. al. algorithm. In the worst case it requires *Number of Users* computations of a pseudo-random function plus decryption of the encrypted attribute value, whereas the worst case scenario for our approach supposes *Number of Users* decryptions of the EPC number plus decryption of the encrypted attribute value.

One issue remains to be dealt with: The ASP still can identify the owners of the submitted values by being able to track the origin of an insert or update query. To prevent it the supply chain participants may deploy an onion routing network [6] (also known as Tor), which will make the communication between the supply chain participants and the ASP anonymous. Such network is relatively easy to deploy and maintain and the only negative side-effect is minor communication latencies. However, since the anonymity is required only for inserts or updates, requests for EPC-related data, which will constitute most of the traffic, can still be efficiently carried out using conventional routing protocols.

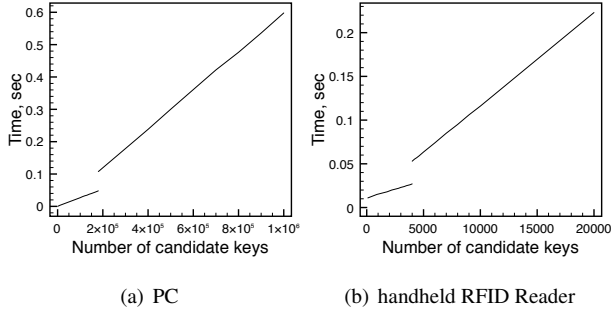


Figure 2. Time Elapsed for  $N$  Decryptions

### 4.3 Performance tests

The average time a user needs to process a tuple with EPC-related data is proportional to the joint number of the owned candidate keys for all protected attribute values she wants to read. To estimate possible delays we simulated the permission verification process for an attribute value and measured the average access time. The tests were run on a PC that may be used to control several stationary RIFD readers and on a handheld RFID reader that has sufficient computational power to work with cryptographic primitives. The PC had Pentium IV 2.80 GHz CPU with 1024 MB of RAM and was running Windows XP; as a handheld RFID reader we used Nordic ID PL3000 supplied with Sharp 200 MHz ARM processor and 32 MB RAM running Windows Mobile 4.0.

Each test run comprised of  $N$  decryptions of a 128-bit ciphertext block. Each decryption was performed with one of the candidate keys and the result of each decryption was compared with the valid plaintext. As the encryption algorithm we used AES cipher in CTR mode with 128-bit key. The test application was written in C and used LibTomCrypt cryptographic library for AES implementation. The test results are presented at Figure 2.

The change of the slope on both diagrams is caused by different modes of operation of the search procedure. The gentle slope corresponds to the mode where  $N$  cipher variables were initialized with candidate keys and then sequentially used in a cycle where the ciphertext containing the EPC number was decrypted and verified against the valid EPC number. The steep slope corresponds to the mode where a single cipher variable was reinitialized with a new key at each iteration of the cycle. Since the preinitialization of the cipher variables has to be performed only once, in the former mode the search was performed more than twice as fast as in the latter. To avoid I/O delays the preinitialized cipher variables had to fit into RAM of the device, thus making it necessary to switch to the second mode when  $N$  was getting too large: The PC with 780 MB of RAM and the RFID-reader with 19 MB of RAM available for the test

application could process in the first mode about 180000 and about 5000 candidate keys correspondingly.

Let's interpret these results for a hypothetical supply chain that consists of  $S$  stages and at each  $i$ th stage operate  $N_i$  participants. A participant operating at  $i$ th stage submits values to a number of attributes,  $M_i$  of which are assigned some access permissions. For example, the table displayed at Figure 3 may correspond to a supply chain with  $S = 2$ ,  $N_1 = N_2 = 3$ ,  $M_1 = 2$ ,  $M_2 = 3$  (the protected attributes are grayed). In order to check access permissions for an attribute value submitted at  $i$ th stage at most  $N_i$  keys have to be tried out. To check access permissions for attribute values that are defined at the same stage (e.g. `manufacturer`, `manufact_date`) it is sufficient to determine the key only for one of them since the rest have the same owner. So, if there is a supply chain participant with full read access to the EPC-related data, to read values of all protected attributes of a tuple at most  $N_1 + \dots + N_S$  keys will have to be tried.

| 1st manufacturer<br>2nd manufacturer<br>3rd manufacturer |          |                      |                      | 1st retailer<br>2nd retailer<br>3rd retailer |                      |                      |
|--|----------|----------------------|----------------------|--|----------------------|----------------------|
| EPC  | name     | manufact_date        | manufacturer         | sale_date                                    | ext_guarantee        | return_reason        |
| $x_{11}$   | $x_{12}$ | $E_{k_{23}}(x_{13})$ | $E_{k_{24}}(x_{14})$ | $E_{k_{35}}(x_{15})$                         | $E_{k_{36}}(x_{16})$ | $E_{k_{37}}(x_{17})$ |
| $x_{21}$   | $x_{22}$ | $E_{k_{13}}(x_{23})$ | $E_{k_{14}}(x_{24})$ | $E_{k_{15}}(x_{25})$                         | $E_{k_{16}}(x_{26})$ | $E_{k_{17}}(x_{27})$ |
| ...  |          |                      |                      |  |                      |                      |
| $x_{n1}$   | $x_{n2}$ | $E_{k_{33}}(x_{n3})$ | $E_{k_{34}}(x_{n4})$ | $E_{k_{25}}(x_{n5})$                         | $E_{k_{26}}(x_{n6})$ | $E_{k_{27}}(x_{n7})$ |

Figure 3. EPC-related Data

To put some real numbers behind this hypothetical scenario we consider a supply chain of major european apparel producer Gerry Weber AG. The supply chain consists of about 3000 participants and is structurally presented at Figure 4 (the information was obtained in an interview).



Figure 4. Structure and Numbers of Participants in Supply Chain of Gerry Weber AG

Thus  $S = 7$ ,  $N_1 + \dots + N_7 \approx 2811$ . Therefore, to fully read a tuple with EPC-related data a supply chain participant with full read access in the worst case will have to try about 2811 keys and about 1406 keys on average. According to the results of the performance tests, when using analogous equipment on average it will take about 0.4 and 16 milliseconds on the PC and the handheld RFID reader correspondingly. So, when all the computations are performed on the handheld reader, accessing the EPC-related

data for 60 simultaneously read tags will take about one second whether at the same time interval the PC can process about 2500 tags what allows to use it as a controller for several stationary readers.

#### 4.4 Write Access Control

So far we have only described the solution that allows to control read access permissions. But often it is also necessary to be able to protect the outsourced data from unauthorized changes. Since the users have no physical access to the database, it should be at least possible to detect such changes. A standard technique for verifying integrity of a value is to digitally sign it. Let  $(\mathcal{G}, \mathcal{E}, \mathcal{D})$  be an asymmetric encryption scheme where  $\mathcal{G}$  is a key generation algorithm,  $\mathcal{E}$  encryption and  $\mathcal{D}$  decryption algorithms. Algorithm  $\mathcal{G}$  generates private key  $s$  that is used for signing and public key  $v$  that is used for signature verification. To protect value  $x$  from an unauthorized modification its owner computes digest  $d = h(x)$  where  $h$  is a cryptographically strong hash function and produces a digital signature by encrypting the digest using encryption algorithm  $\mathcal{E}$  and private key  $s$ . The signature is appended to the value:  $(x|sig) = (x|\mathcal{E}_s(h(x)))$ . To verify the integrity of the value one should compute its digest, decrypt the signature with public key  $v$  and compare the result of the decryption with the computed digest:  $d(x) \stackrel{?}{=} \mathcal{D}_v(sig)$ . The cryptographic strength of the hash function implies the infeasibility of forging a message with the same digest and the fact that key  $s$  is known only to the owner of the value ensures that it was she who computed the signature. Thus, the integrity of the value is confirmed when the equality holds true.

For the outsourced database it is also important to fix the position of the value in the table. If the digital signature is based only on the value, a malicious user may unnoticeable interchange values of tuples or attributes, for example, substituting the manufacturing dates of product with expired shelf life with the manufacturing date of a product that is still useable. In order to prevent such modifications, the digest should be computed from a combination of the value, its attribute name and corresponding EPC number:  $h(epc|attr_j|x)$ .

Thus, a value of attribute  $attr_j$  that is owned by user  $U_i$  and is protected from unauthorized read/write has to be stored in the following form:  $(E_{k_{ij}}(epc_{padded}|x)|\mathcal{E}_{s_i}(h(epc|attr_j|x)))$ . Key  $v_i$  is made publicly available and key  $k_{ij}$  is distributed to the users with read permissions for this value.

## 5 Conclusion

In this paper we presented a technique that allows supply chain partners to define read/write access permissions

to the owned portions of EPC-related data stored in the central repository. The technique allows to dynamically grant read access permissions to users without the necessity to re-encrypt the data, does not impose any communication overhead, is easy to implement and is lightweight enough to be supported by a handheld RFID reader. The results of the performance tests show that when used on a handheld RFID reader or a PC that controls several stationary RFID readers the approach is efficient enough to be applicable to a supply chain consisting of several thousands participants.

## References

- [1] S. G. Akl and P. D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.*, 1(3):239–248, 1983.
- [2] A.T. Kearney. [www.atkearney.com](http://www.atkearney.com).
- [3] F.-E. Bertino and F.-R. Sandhu. Database security-concepts, approaches, and challenges. *IEEE Trans. Dependable Secur. Comput.*, 2(1):2–19, 2005.
- [4] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, 2006.
- [5] E. Damiani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Key management for multi-user encrypted databases. In *Proceedings of the 2005 ACM workshop on Storage security and survivability*.
- [6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.
- [7] EPCglobal. EPCglobal Tag Data Standards Version 1.3. March 2006.
- [8] S. Evdokimov and O. Günther. Encryption techniques for secure database outsourcing. In *Proceedings of 12th European Symposium On Research In Computer Security*, 2007.
- [9] H. Hacıgümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over Encrypted Data in the Database-Service-Provider Model. In *Proceedings of the 28th SIGMOD Conference on the Management of Data*. ACM, 2002.
- [10] M.-S. Hwang and W.-P. Yang. Controlling access in large partially ordered hierarchies using cryptographic keys. *J. Syst. Softw.*, 67(2):99–107, 2003.
- [11] S. J. MacKinnon, P. D. Taylor, H. Meijer, and S. G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Trans. Comput.*, 34(9):797–802, 1985.
- [12] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill Science/Engineering/Math, 2002.
- [13] D. X. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy*, 2000.
- [14] Y. Tang. Sharing session keys in encrypted databases. In *ICEBE '06: Proceedings of the IEEE International Conference on e-Business Engineering*, 2006.