Enabling Secure Data Exchange

Gerome Miklau
University of Washington
gerome@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

1 Introduction

The emergence of diverse networked data sources has created new opportunities for the sharing and exchange of data. In support of this, a fruitful line of research has resulted in distributed data processing and integration systems [19, 17, 29, 30, 3]. However in practice, fear of unauthorized disclosure or malicious tampering requires that data stay safely behind firewalls or remain protected by secure servers. Our goal is to overcome these limitations and enable secure data exchange and sharing in distributed integration scenarios. Such scenarios are characterized by many interacting data sources and many data consumers. Primary sources create and publish data; intermediate sources combine, extract, and modify the data for further dissemination; data consumers query it. This paper describes issues in secure data exchange, and illustrates some solutions proposed in the authors' own work.

The basic requirements of secure data exchange are *confidentiality* and *integrity*. Confidentiality means that unauthorized parties are prevented from *reading* data. In data exchange, confidentiality is provided through encryption and managing keys that allow access. Confidentiality benefits data sources who need to protect data. Integrity (in its basic form) means that unauthorized parties are prevented from *modifying* data. In data exchange, integrity is provided through digital signatures and data certification techniques. Integrity benefits both data sources (who need to make sure data attributed to them is not modified) and data consumers (who need guarantees that the data they use has not been tampered with).

Confidentiality and integrity are distinct goals and the tools for each are different. In particular, techniques for providing confidentiality do not by themselves provide integrity. Participants can guarantee both properties by combining techniques. We describe the basic features of our envisioned framework for secure data exchange below:

The data Base data is annotated with *security metadata*. We use XML data since it is the preferred format for data exchange. For confidentiality, the metadata contains information about access control requirements and encryption algorithm details, while the base data is protected by encryption. For integrity, the security metadata contains evidence of authenticity (in the form of signatures), key references and algorithm descriptions. The resulting data can be exchanged freely, without relying on secure communication channels or secure servers.

Key management Confidentiality of the data depends on the secure distribution of keys to intended recipients, while verification of integrity depends on authentic retrieval of a data source's public keys. From a

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

pessimistic point of view, it may seem we have merely transferred problems of confidentiality and integrity from the data onto the keys. For instance, without a signature the data consumer may doubt the authenticity of the data. With a signature the data consumer can trust the data, but only if she trusts the source providing the public key. But, as many cryptographic protocols have proven, there is practical value in consolidating the challenges of ensuring confidentiality and authenticity on a few keys rather than on the data itself [28]. The techniques used to secure keys (like public key infrastructures and trusted certificate authorities) are definitely relevant to secure data exchange but are out of our scope here. We assume the parties involved employ these techniques and focus instead on the range of challenges that remain.

Processing the data Intermediate sources and data consumers process the data by applying keys to decrypt the data, by verifying integrity of the data by evaluating digital signatures, and by computing query results over base data (although these are not necessarily performed in distinct stages). In addition, sources construct new data with proper confidentiality and integrity properties by encrypting and signing.

In the next section we motivate the goal of secure data exchange with three applications, highlighting the security requirements of each. We discuss confidentiality in Section 3 and integrity in Section 4. We identify research challenges in Section 5 and then conclude.

2 Applications of Secure Data Exchange

Scientific data management, e-business, and personal identity databases are each compelling applications for secure data exchange.

Scientific data exchange As a representative scientific domain we consider the field of molecular biology. From a few primary sources, containing original experimental data, hundreds of secondary biological sources [2] are derived. The secondary sources export views over primary sources and/or other secondary sources, and usually add their own curatorial comments and modifications. These databases are often published on the Web as large XML documents – not stored in proprietary systems or servers that can provide security guarantees. The data consumers are scientists, and a significant fraction of research takes place in so-called "dry" laboratories using data collected and curated by others.

The risk of malicious tampering with the data is usually not the primary concern in this setting. Instead, the main issues are attributing and retaining authorship and avoiding the careless modification of data through integrity controls. In addition, although many of these databases are released to the public, some forms of confidentiality may nevertheless be important in some cases: when scientific data includes fields that identify individuals, when the data owners wish to audit their data usage by requiring keys for access, or when scientists wish to temporarily limit the release of raw experimental data.

E-business data exchange Today's business transactions very often require data exchange across corporate organizational boundaries, between many parties, and are increasingly performed by machines unattended by humans. The data exchanged may contain financial terms, detailed technical data about products, proprietary corporate information, or personal information about customers. Both confidentiality and integrity of e-business data exchange are critical for ensuring business continuity, compliance with regulations, and protection of corporate assets.

It is worth noting that while there are well-developed technologies for authenticated, confidential point-to-point messaging between two counter-parties in a transaction, that authentication usually happens once for all transmitted data. Even if the authentication information is recorded, often it is not associated directly with the data, or cannot be associated with specific parts of the data.

Personal identity databases A large class of databases, which we call "personal identity databases", have in common the fact that they contain personally identifying information about individuals (e.g. census data, medical databases, organizations' member lists, business customer data). Such databases can be viewed as intermediate sources that collect data from primary source individuals who donate their personal data. The secondary sources may disseminate or further integrate this identity data.

For example, individuals present their personal data to a hospital database when admitted. Hospitals add treatment data and send patient records to an insurance company that integrates it with data of patients from other hospitals. Due to privacy regulations, confidentiality of such databases is critical. But integrity is equally important. If an individual applies for insurance coverage, the insurance company will evaluate the cost of insuring the individual based on the data in its database. An individual should have the right to verify the accuracy of that data. This can be accomplished if the data carries integrity metadata, and the insurance company is required to present the data and its evidence of integrity. To continue the example, some of the individual's personal data will be signed by the individual, some will be signed by individual's doctors etc. Regulations need to be in place that make it illegal for the insurance company to base coverage decisions on data that is not checked for integrity.

3 Providing Confidentiality

Confidentiality is an assurance that unauthorized parties are prevented from accessing data. For exchanged data, confidentiality is provided by encryption. A few recent projects [5, 4, 25] have encouraged data sharing and exchange by allowing a single published XML document to ensure confidentiality for authors in the context of a set of data consumers with different access rights. We elaborate below on some distinguishing features of our framework [25], and also mention some of our recent work on information disclosure in data exchange [26].

3.1 Confidentiality for Published Data

We have developed a framework that allows a data owner to restrict access to published XML data through encryption. The XML document is partially encrypted (and sometimes super-encrypted) to support a declared access control policy. Then the data owner publishes it on the Web, and anyone can download it, process it, and/or re-publish it or fragments thereof. The encryption ensures that only users having specific keys can access restricted data. In addition, we enforce *conditional* access to data, granting access to users who can supply certain data values contained in the database. Conceptually this is related to binding pattern limitations on a query interface. This feature may, for example, be used to permit any medical professional who knows a patient's social security number and date of birth to access certain fields of that patient's medical record. The main components of our framework are the following:

1. A query language for expressing policies – The data owner expresses access control policies using extensions to XQuery. For example consider the following policy query:

   ```
   SUFFICIENT
   FOR       $x in /doc/subjects/subject
             $y in /doc/psychs/psych
   WHERE     $x/examining-psych/id = $y/id
   KEY       getKey( $y ) keyChain("psych")
   TARGET    $x
   ```

   It specifies that a psychologist examiner is allowed to see all subjects he examined. A new key is generated by the function `getKey( $y )` for each psychologist `$y` (or retrieved, if it already exists), in the key chain named "`psych`", and that key grants access to all subjects examined by that psychologist.

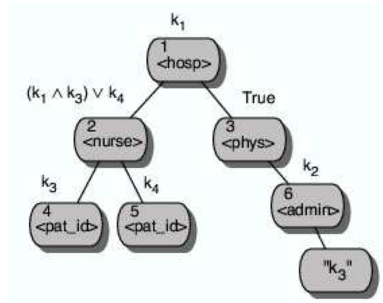

Figure 1: A tree protection.

Notice that if a subject was examined by multiple psychologists[1] then each of them has access to that subject. In addition, other policy queries may impose other restrictions to subject elements, to their sub-elements, or to their parents.

2. A logical model for protecting XML document – We describe a logical model for protecting an XML document tree, in which nodes are labeled with logical expressions over keys. An example is shown in Fig 1. Access to a subtree requires possession of keys that satisfy the logical expression "guarding" the subtree. For example, a user holding the keys k_1 and k_4 will have access to the nodes 1, 2, 3, and 5, and to no other node. If, in addition, the user holds key k_2, then she can unlock node 6, then access its child and acquire key k_3: this in turn gives her access to node 4.

   All policy queries that the data owner specifies are executed over the XML document and result in one logical protection of the XML document. This protection model enables rewriting and optimization at the logical level (to avoid size blow-up due to super-encryption), before the protected tree is represented physically as a partially-encrypted document.

3. Techniques for generating protected XML – Once the logical protection is optimized, it is used to generate a single partially-encrypted XML document that enforces the access control policies. In [25] we validate experimentally the efficiency of generating such XML documents, as well as the positive impact of the logical optimizations and other improvements like compressing data before encrypting.

3.2 Information Disclosure in Data Exchange

A common approach to hiding confidential data in databases is to create views that simply omit the secret items. This is also applied in data sharing. The data owner decides what views to share with each user, then uses physical protection mechanisms to enforce that each user gets only the views he is entitled to. However further dissemination of these views, or collusion between users, may compromise confidentiality. While nothing can be done after the fact, the data owner may wish to understand the amount of confidential information that leaks under different collusion scenarios. To address this problem we have studied in [26] the *query-view security problem*: given a set of views $V_1, \ldots, V_n$ and a secret query S, do the views leak any information about the secret query? To formalize information leakage we have adapted Shannon's definition of perfect secrecy: the query S is secure w.r.t. the views $V_1, \ldots, V_n$ if the probability of an attacker guessing the answer to the secret query remains the same before and after learning the view answers. Among other results, we have shown that it is possible to decide query-view secrecy for the case of conjunctive queries, and that the query complexity is Π_2^p-complete.

[1] This happens when subject has more than one examining-psych sub-elements.

4 Providing Integrity

Integrity, in its most basic form, is an assurance that unauthorized parties are prevented from modifying data. Unauthorized modification of data may be malicious, or simply careless. Integrity is more complex than confidentiality, and we distinguish between *data integrity* and *operation integrity* below. We describe here the integrity problem in data exchange, mention some of the existing techniques, and describe some future directions we hope to pursue in supporting integrity properties.

Data Integrity Data integrity can include one or more of the related properties of *origin authenticity*, *temporal commitment*, *non-repudiation*, and *freshness*. Origin authenticity is an assurance that the data comes from the attributed source (this implies that in addition it has not been modified from its original state). Temporal commitment is an assurance that the data was not modified after the time of receipt of signature. Non-repudiation is an assurance that the source of the data cannot deny authorship, while freshness is an assurance that the data is not a copy of data signed by the source, cached by an unauthorized party, and later presented as authentic. (This is commonly called a replay attack [20].)

Data integrity is provided using digital signatures, usually based on public key cryptography [20]. Any digital signature can provide origin authenticity and temporal commitment, the first two properties above. Additional features can be added to a digital signature to provide properties of non-repudiation and freshness as well. An XML Recommendation [15] describes a schema for signing data and representing signature metadata, and recent work has integrated signatures into XML processing [7, 6].

Query integrity Query integrity is an assurance that a query over a database has been performed accurately, which is a consideration distinct from the integrity of the individual data elements returned by the query. For example, data integrity techniques may make it difficult for the source to falsely introduce unauthorized data, but the source may still omit pieces of data or compute query incorrectly.

Research into consistent query protocols [18, 27, 13, 12] can provide query integrity in some cases. The techniques are based on Merkle trees [21, 22] and allow signing of a database D such that given a query Q and an possible answer x to Q, a verification object can be constructed which proves that $x = Q(D)$. This means that if a database is signed, then for some queries (like simple selections or range queries) it is possible to provide evidence of query integrity, relative to a prior signature over the database. The important point here is that the entire database is signed once, and this single certificate can later be used to verify the integrity of any query. It should be noted that there are limitations to the efficiency and privacy of these techniques – the verification objects can be large, and may reveal data elements that are not in the query output.

4.1 Managing integrity

Consider the simplest data exchange scenario: a transformation is applied to a data source to compute a view, which is exchanged with a partner. The integrity properties of the input must be propagated through the query or view to the output. This is related to some extent with managing data provenance [9, 10, 11, 8] which involves tracing and recording the origin of data and its movement among databases. In our case, however, the "integrity provenance" constitutes hard-to-forge evidence of origin (i.e. digital signatures) rather than a mere claim of origin. In data exchange the transformation querys are repeated and amplified many times. The main types of querys that impact the management of integrity provenance are:

Object fusion We construct a new object by integrating data from two different sources. For example a source such as `dblp` provides bibliographic information about a publication, which we merge with information about the number of citations obtained from, say, `citeseer`. The new object needs to carry integrity provenance that specifies that it was obtained by combining data from both sources.

Choice We wish to construct a new piece of data, based on evidence found at two (or more) different data sources. Each piece of evidence is sufficient by itself to support our new data item. We need to be able to express this as integrity provenance. Notice how this differs from object fusion: there a user needs to trust *both* sources in order to trust the fused object, while here it suffices if she trusts only one source.

Extraction We wish to publish a piece of data that is a fragment of a larger item which has a signature. Here the integrity provenance needs to include the signature on the larger object, and also an indication of how the fragment was extracted. Merkle trees address a special case of extraction, when a single tuple is extracted from a certified database.

Combination A new piece of data is constructed by combining together several data items collected from several sources. In addition to the integrity provenance of each item, now we need to include the signature of the author who performs the combination.

4.2 Querying integrity provenance

Integrity provenance can interact with a query language in different ways. First, the query language may require users to place restrictions on the data items sought, based on their integrity provenance. For example, the user may want to specify that she trusts only sources S_1, S_2, S_3, and ask the system to retrieve only data items that can be fully certified solely by these sources. Second, when the data is transformed or integrated with a query language, the integrity provenance information needs to be propagated in the output data. As suggested above, the meaning of this propagation is not obvious. One possibility is to allow users to specify policies on how the propagation is being performed. Consider the union of non-disjoint data elements. A policy may require each signature to be trusted (corresponding to logical AND), or may simply require the presence of one trusted signature (logical OR), or may always prefer one signature (when available) over another. Third, once a query is evaluated, the user may request a proof of the validity of the answer. This proof may range from a simple enumeration of all signatures used in the data items returned, to a complex expression indentifying all signatures that have been used in different parts of the data source that affected the query's answer.

5 Research Challenges

Data enriched with confidentiality and integrity metadata poses some special challenges, enumerated below.

Query language challenges Physically, the data and the security metadata may often be stored together. But logically, they should be separated. A query language should have abstractions that allow users to refer to the security metadata separately from the data. Users should never have to express queries over the mixed schema. In addition, query answers may need to contain explanations of their result since, for example, the answer to a query could be empty because evaluation on the base data resulted in an empty answer, or because the confidentiality or integrity conditions were not met.

Query processing challenges Managing the metadata together with the data during query processing will pose challenges for the system. Some accesses to the metadata will be cheap, but most often they require expensive decryption or signature verification. The query planner should favor plans that defer these expensive operations as much as possible. There are some connections here to research into query execution with expensive predicates [16], in which the customary pushing of selections is not always beneficial.

Schema challenges The base data exchanged in our scenarios will usually conform to some agreed-upon schema. However, as we have described, the base data may be transformed into partially-encrypted elements, signatures may be "wrapped" around the base data, or both. While XML schema have been defined for encryption [14] and digital signatures [15], the secured data instances will not generally conform to the base data schema or the security metadata schema. This complicates querying and validating of the data, as well as schema evolution.

User assistance and deployment In order to generate data instances with proper security metadata it is essential to have machine assistance. Signatures and encryption keys consist of random strings that cannot be manipulated in any reasonable way by humans. While a scientist may have used a text editor in the past to update a data entry, now that scientist needs more advanced tools to add signatures and perform encryption. Users also need "key chains" to manage keys given to them or granted by them.

Proving security Claims of confidentiality and integrity require formal proofs of security. Since our data instances result from the application and combination of cryptographic primitives, proofs of security are hard to generate. In the case of confidentiality, we described a logical language for protected XML trees and a process for generating encrypted data from the logical model. Ideally, we would like proofs of security to follow from the soundness of a construction in the logical model. Recent work [1, 23] from the cryptographic community addresses precisely this goal and needs to be extended and applied to our setting.

6 Concluding Remarks

Providing confidentiality and integrity is critical to facilitating sharing and exchange of data. Since data is exchanged beyond domains of influence of data authors, we can't depend on secure systems to enforce confidentiality and integrity, but must rely on techniques of cryptography. Yet, while there are many compelling cryptographic primitives available, applying and adapting them to complex data management is a major challenge. For one, in data exchange, we are concerned not with blocks of data or messages treated in isolation, but with databases which are structured collections of elements, to which confidentiality and integrity may apply at various levels of granularity. Second, communication is not point-to-point but involves many parties in a complex network as described in our application scenarios. We have described our view of the key problems, referred to existing work, and proposed future directions. The interested reader may consult [24] for a more detailed version of this paper.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP Int. Conference on Theoretical Computer Science*, Sendai, Japan, 2000.

[2] Andreas D. Baxevanis. Molecular biology database collection. Nucleic Acids Research, available at www3.oup.co.uk/nar/database/, 2003.

[3] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The hyperion project: from data integration to data coordination. *SIGMOD Rec.*, 32(3):53–58, 2003.

[4] E. Bertino, B. Carminati, and E. Ferrari. A temporal key management scheme for secure broadcasting of xml documents. In *Conference on Computer and communications security*, pages 31–40. ACM Press, 2002.

[5] E. Bertino, S. Castano, and E. Ferrari. Securing xml documents with author-x. *IEEE Internet Computing*, 5(3):21–31, 2001.

[6] E. Bertino, G. Mella, G. Correndo, and E. Ferrari. An infrastructure for managing secure update operations on xml data. In *Symposium on Access control models and technologies*, pages 110–122. ACM Press, 2003.

[7] L. Bull, P. Stanski, and D. M. Squire. Content extraction signatures using xml digital signatures and custom transforms on-demand. In *Conference on World Wide Web*, pages 170–177. ACM Press, 2003.

[8] P. Buneman. Curated databases, November 2003. personal communication.

[9] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[10] P. Buneman, S. Khanna, and W. C. Tan. On propagation of deletions and annotations through views. In *PODS '02*, pages 150–158, 2002.

[11] Y. Cui and J. Widom. Practical lineage tracing in data warehouses. In *ICDE*, pages 367–378, 2000.

[12] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. G. Stubblebine. Flexible authentication of xml documents. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 136–145. ACM Press, 2001.

[13] P. T. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic third-party data publication. In *IFIP Workshop on Database Security*, pages 101–112, 2000.

[14] D. Eastlake and J. Reagle. Xml encryption syntax and processing. http://www.w3.org/TR/xmlenc-core, 3 October 2002. W3C Proposed Recommendation.

[15] D. Eastlake, J. Reagle, and D. Solo. Xml signature syntax and processing. http://www.w3.org/TR/xmldsig-core, February 12 2002. W3C Recommendation.

[16] J. M. Hellerstein and M. Stonebraker. Predicate migration: Optimizing queries with expensive predicates. In *SIGMOD Conference*, pages 267–276, 1993.

[17] Z. G. Ives, A. Y. Levy, J. Madhavan, R. Pottinger, S. Saroiu, I. Tatarinov, S. Betzler, Q. Chen, E. Jaslikowska, J. Su, and W. Yeung. Self-organizing data sharing communities with SAGRES. In *SIGMOD '00*, page 582, 2000.

[18] J. Killian. Efficiently committing to databases. Technical report, NEC Research Institute, February 1998.

[19] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 251–262, Bombay, India, 1996. VLDB Endowment, Saratoga, Calif.

[20] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[21] R. C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[22] R. C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[23] D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.

[24] G. Miklau. Research problems in secure data exchange. University of Washington Technical Report 04-03-01, March 2003. Available at www.cs.washington.edu/homes/gerome.

[25] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB*, pages 898–909, September 2003.

[26] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. to appear SIGMOD '04, June 2004.

[27] R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs on a committed database.

[28] B. Schneier. *Applied Cryptography, Second Edition*. John Wiley and Sons, Inc., 1996.

[29] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 1996.

[30] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. L. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *SIGMOD Rec.*, 32(3):47–52, 2003.