Managing Integrity for Data Exchanged on the Web

Gerome Miklau Dan Suciu
University of Washington
{gerome, suciu}@cs.washington.edu

ABSTRACT

The World Wide Web is a medium for publishing data used by collaborating groups and communities of shared interest. This paper proposes mechanisms to support the accuracy and authenticity of published data. In our framework, publishers annotate data with virtually unforgeable evidence of authorship. Intermediaries may query, restructure, and integrate this data while propagating the annotations. Final recipients of the data may then derive useful conclusions about the authenticity of the data they receive.

1. INTRODUCTION

The emergence of diverse networked data sources has created new opportunities for the sharing and exchange of data. In particular, the Web has become a medium for publishing data used by collaborating groups and communities of shared interest. Once published, it is common for other parties to combine, transform, or modify the data, and then republish it.

In such distributed settings there are few mechanisms to support users in trusting the accuracy and authenticity of data they receive from others. To address this problem, we investigate guarantees of *data integrity* for exchanged data. Integrity is an assurance that unauthorized parties are prevented from modifying data[1]. Integrity benefits both the authors of data (who need to make sure data attributed to them is not modified) and the consumers of data (who need guarantees that the data they use has not been tampered-with).

After publication, the owner of data can never directly prevent modification of the published data by recipients. But it *is* possible to annotate published data with virtually unforgeable evidence of its authenticity that can be verified by recipients. Data authors need techniques which allow them to annotate data with claims of authenticity. These claims should be difficult to forge or transfer, and must be carried along with the data as it is exchanged and transformed. Subsequent users must then be able to derive useful integrity guarantees from query results containing these claims. We explore here techniques to accomplish these goals.

To illustrate the importance of integrity in data exchange, we describe two applications: scientific data exchange and personal identity databases.

Scientific data exchange. As a representative scientific domain we consider the field of molecular biology. From 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 [23]. These databases are often published on the Web, as structured text files – 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. An illustration of this scenario is provided in Figure 1.

The threat of malicious tampering with the data is usually not a primary security concern in this setting. Instead, the main issues are attributing and retaining authorship and avoiding the careless modification of data. To the best of our knowledge, security properties are rarely provided in scientific data exchange. Although in some cases authorship is traced, there is little evidence or verification of authorship.

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 donating 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.

Integrity is critical as the data migrates between organizations: only authorized parties should be able to modify data, or create new records, and those parties should be identifiable after the fact. If an individual

[1]We adopt the meaning of integrity common to information security (not databases).

Copyright is held by the author/owner. Eighth International Workshop on the Web and Databases (WebDB 2005), June 16-17, 2005, Baltimore, Maryland.
.

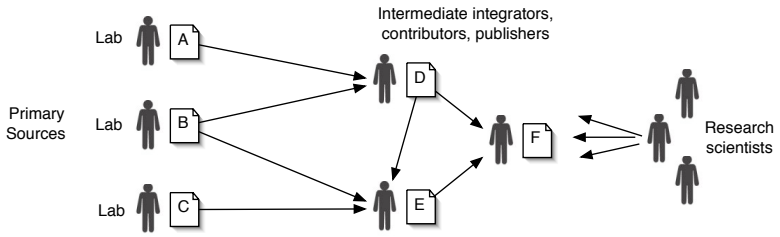

Figure 1: Data publishing and exchange scenarios for scientific data management.

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 himself, 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 verifiably authentic.

Origin authenticity and Origin-critical data

Our particular focus is an aspect of data integrity called *origin authenticity*: an assurance that data comes from an attributed source (and that it has not been modified from its original state) [16].

There are two important threats to origin authenticity. The first is the threat of copying data published by an author Alice. For example, an adversary, Mallory, may duplicate the data received from Alice, remove the original evidence of attribution and claim himself as the author. This threat, while important, is not our focus here. (It requires substantially different techniques like watermarking [1], or legal measures [13].)

Instead we focus on the threat of an adversary tampering with data authored by Alice. This can happen in two related ways. Alice may author some data, which is properly attributed to her, but Mallory changes the data while keeping the attribution. Or more directly, Mallory forges the attribution itself, applying it to data of his choosing. To justify our focus on this threat, we describe next *origin-critical data*, for which tampering is the primary concern.

Origin-critical data is data whose value or utility depends critically on its authenticity or the authority of its source[2]. A table of stock recommendations like {(IBM, buy), (MSFT, sell)} is an example of origin-critical data. The raw data can easily be fabricated or duplicated and therefore is worth little if its source is unknown or the claim of its origin is untrusted. For instance, proof that the author of the data is an expert equity analyst makes the data valuable. If authored by a high-school investment club however, the stock ratings are substantially less useful. Therefore, its origin is critically important. An mp3 file is an example of data whose origin is not critical. The utility of the mp3 file seems to consist solely in the contents of the file. Its source, and the authenticity of its attributed source, are usually not relevant to the listener who is happy to download the file from an unknown party on the Internet.

Many kinds of digital data are origin-critical, and a primary integrity concern is to maintain and manage verifiable claims of authorship. For example, in scientific data management the consumers of data (the scientists) are reluctant to use data that does not come from a reputable source. In e-business transactions, a party may not be willing to act on data received if it is not verifiably authentic. For origin-critical data it is usually in the interest of the participants to retain evidence of origin authenticity.

Our goal is therefore to develop a framework to (1) allow authors to annotate data with evidence of authorship, (2) allow recipients to query, restructure, and integrate this data while propagating the evidence, and (3) enable recipients to derive useful conclusions about the authenticity of the data they receive. In support of these goals, first a pair of integrity annotations are proposed, which are applied to data to represent useful claims of origin authenticity. Then cryptographic techniques are described that can be used to support these annotations so that the claims are not forgeable or transferable. Finally, we assess the requirements and challenges of managing data with integrity annotations.

2. INTEGRITY ANNOTATIONS

Annotations are applied to fragments of a database. For relational data, a fragment may be an individual attribute, a tuple, or a set of tuples. For XML data, a fragment may be a complex subtree, or a forest of subtrees. We propose in this section two related forms of annotation – signature and citation – which are used by data authors to represent claims of origin authenticity. We describe the semantics of these annotations and their use. In the next section we describe cryptographic techniques to implement these forms of annotation.

Signatures. On paper, signatures are intended as proof of authorship or an indication of agreement with the contents of a document. Signatures serve their purpose because they have some or all of the following properties [27]: the signature is unforgeable, the signature cannot be reused on another document, and the signed document is unalterable. For the present discussion we will assume a basic signature primitive possessing these

[2]Some of these ideas were inspired by [5].

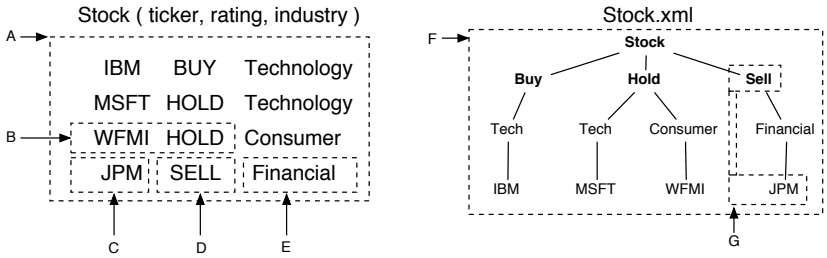

Figure 2: A relational table of stock recommendations (left), the same data represented as XML (right), and an illustration of fragments of the data to be signed.

properties that can be applied to any fragment of data. Realization of the signature primitive is discussed in the next section.

The author signs data to ensure that others cannot modify it. The granularity of signatures can vary: an author can sign an entire table, a tuple, a single column value.[3] Usually signatures are used to associate some data in an unmodifiable way, as shown in the next example. In what follows we use stock recommendations as a simple example of origin critical data, however the intended application domains remain those described in Section 1.

Example 2.1 Figure 2 shows stock recommendations represented as a relational table Stock(ticker, rating, industry) and as an XML document. The dotted regions illustrate portions of the data that are signed, called the target fragment of a signature. Signature $sig(A)$ is applied to target A, i.e. the entire table, and $sig(F)$ is similarly applied to the entire document. If the user wanted to compare the performance of the recommended portfolio represented by Stock, then these signatures provide integrity: poorly performing stocks cannot be removed and outperforming stocks cannot be added after signing. Signature $sig(B)$ and $sig(G)$ are applied to ticker-rating pairs. This associates the ticker name with the rating in an unmodifiable way, however a collection of such signed tuples does not prevent deletions, rearranging or additions to the collection. Signatures $sig(C)$, $sig(D)$, and $sig(E)$ are applied to individual attribute targets. By themselves, these three are probably not useful signatures since they do not authenticate the association between ticker and rating, which is of primary importance here.

The choice of signature granularity is application dependent. The signature of an entire database protects against all possible changes, but may be inefficient since verification must be performed on the entire database. In practice authors sometimes want to authorize smaller pieces of data. In many contexts, the author may wish to publish data signed in more than one way, with varying granularity. This allows a recipient to republish the data in various forms, retaining its evidence of authenticity. For example:

1. In Example 2.1, an author may wish to sign all subsets of tuples by sector, so that the data consumer can extract authenticated data for any relevant industry sectors, and omit others.
2. Consider a college transcript represented as a structured document, and signed by a academic administrator. The original form of the transcript may include fields the student wishes to hide when the transcript is submitted to a potential employer (e.g. date of birth). The administrator may wish to provide two signed versions of the document – one with the date of birth, and one without. (The administrator would not however, want to grant the student signed versions of the transcript that omit bad grades.)
3. If the order of elements is critical for an XML document, then a signature must secure the order. For other data, the author may wish to sign an unordered collection, allowing any ordering to be authenticated. In this case, the author would need to sign all possible orderings, or use a signature primitive that is order insensitive (we return to this in the next section).

[3]Although the main focus for data exchange is XML, we present some examples in relational form for simplicity of presentation.

Citations. We propose another integrity annotation that allows for the *citation* of signed data. We define a citation to be an annotation of a data fragment (the derived data), with a query and a reference to some signed data (the target). A citation represents a claim of authenticity: the derived data is the result of evaluating the query on the target fragment. The following examples provide some intuition, and an illustration of the flexibility of citations.

Example 2.2 Consider again the stock recommendations in Figure 2. Table 1 presents four examples of citations. For each, the first column is a derived fragment (tuples or sets of tuples in this case). The second column is the citation query which is expressed as a conjunctive query over the signed target fragment. Here A refers to the fragment signed by $sig(A)$. The last column indicates that the target is backed by a signature. We describe the meaning of each:

1. Citation (1) has derived data consisting of two tuples. Its citation claims that the fragment is the result of evaluating query C_1 on the target (the entire Stock table) which is signed by signature $sig(A)$.
2. Citation (2) is very similar with a different selection condition in the citation query.

3. Citation (3) consists of the same derived data as (2), however its citation query is different: it claims that the derived data is *contained* in the result of query C_3. Clearly this citation provides a different authenticity guarantee than citation (2).
4. The target data of a citation was signed in each of the examples above, but in other cases may instead be cited, resulting in a composition of citations. Assume that the derived fragment from Citation 1 is called T_1. Citation 4 therefore refers to a fragment that is itself cited. The claim of authenticity here is the composition of the individual claims. That is, the citation claims that tuple (MSFT, HOLD) is the result of query C_4 evaluated on table T_1 which itself is the result of citation query C_1 on the original data signed with $sig(A)$.

Since a citation is merely a claim, it must be verified by checking the signature of the cited source, and verifying that the cite fragment is in fact the result of the citation query evaluated on the citation source. (In the next section we mention techniques to make this verification procedure more efficient.) It is worth noting that a citation is a generalization of a signature. A citation whose query is the identity query is precisely a signature as described above.

Citations are useful because they do not require the compliance of the author, and provide additional flexibility if the signature on the source data does not permit the extraction a user desires. Citations are also useful for representing the relationship of an aggregation to its contributing values. However, citations may not offer the same level of integrity guarantee as a signature, and as the example shows, the same data may be cited using more than one citation query resulting in different authenticity conditions. Notice that Citations (2) and (3), as well as $Sig(B)$ from Example 2.1, are each annotations representing a claim of integrity about the target (WFMI, HOLD). Each of these integrity annotations has a different meaning. The distinction may be important, and careful reasoning about integrity semantics may be required.

3. CRYPTOGRAPHIC TECHNIQUES

Our objective is not merely to carry *claims* of authenticity along with data, but to propagate virtually unforgeable *evidence* of authenticity, verifiable by recipients. To do this we must employ cryptographic techniques. The most basic is the digital signature, which can be used to implement the basic signature annotation above. We then describe more advanced techniques which support extensibility. We show how Merkle hash trees can be used to design a signature primitive permitting controlled removal of elements from a signed collection. We then give an overview of more advanced techniques from the cryptographic literature that can be used to implement extensible signatures and citations.

Digital signatures

Digital signatures [11] are the basic tool for supporting origin authenticity. Digital signature schemes are generally based on public-key cryptography [11], and consist of two operations: signing and verification. Alice signs a piece of data by computing the one-way hash of the data and then encrypting the hash with her private key. The result is the *signature value*, and accompanies the data element when published. RSA [25] and SHA-1 [26] are examples of public key encryption and hash functions, respectively, from which a signature scheme can be built. Bob verifies a signature by retrieving Alice's public key, using it to decrypt the signature, and checking that the result is equal to the hash of the data element purportedly signed.

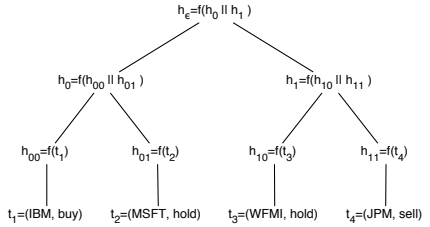

Figure 3: Merkle hash tree over stock tuples.

A verified digital signature provides extremely strong evidence of origin authenticity. The digital signature therefore provides the basic implementation for our signature annotation described above. Publishers generate public/private key pairs, add signature values to their data, and references to public key resources. Recipients verify signatures and propagate signatures to versions they in turn publish (as long as they publish the data in precisely the form it is signed).

Extensible signature techniques

As mentioned above, it is often necessary to sign data in more than one way, or to sign data in such a way that certain modifications are permitted without invalidating the signature. The naive way to support this flexibility is for the author to publish many signatures along with the data. This can be very inefficient since, for example, providing signatures for every possible order in a collection would require an exponential number of signatures. We hope to avoid this by employing the techniques described next.

Merkle trees to allow authorized deletions. Suppose Alice wants to sign a collection of data items so that Bob can delete items but not add new items. If Carol receives a modified collection from Bob, she should be able to verify that each item was indeed authored by Alice, although some items may be missing. Alice's signature permits authorized deletions in this case, and this effect can be implemented using Merkle trees [17]. Note that signing each item in the original collection individually allows unauthorized mixing of items if Alice signs more than one collection over a period of time.

We illustrate how Alice would sign the collection of stock recommendations by building the hash tree illustrated in Fig. 3. Alice uses a collision-resistant hash function f to build a binary tree of hash values as follows. First, she computes the hash for each tuple t_i. Then she pairs these values, computing the hash of their

Table 1: Citations, referring to the relational data in Figure 2.

	Derived fragment	Citation query	Target fragment	Signature / Citation
(1)	(IBM, BUY) (MSFT, HOLD)	$C_1(t,r)$:- A(t, r, "Technology")	A	Sig A
(2)	(WFMI, HOLD)	$C_2(t,r)$:- A(t, r, "Consumer")	A	Sig A
(3)	(WFMI, HOLD)	$C_3(t,r) \subseteq$ A(t, "HOLD", i)	A	Sig A
(4)	(MSFT, HOLD)	$C_4(t,r)$:- T_1(t, "HOLD")	T_1	Cit T_1

concatenation (denoted || in the figure) and storing it as the parent. She continues bottom-up, pairing values and hashing their combination until a root hash value h_ϵ is formed. Note that h_ϵ is a hash value that depends on all tuples in her database. Alice publishes a description of f along with h_ϵ signed with her private key.

If Bob would like to delete t_3 from the collection, he will publish to Carol tuples t_1, t_2, t_4 along with h_{10} and the root hash signed by Alice. Carol will verify the authenticity of the data by recomputing the Merkle tree up to the root hash, and verifying Alice's signature on it. We assume a fixed order for the tuples and some specified structural information allowing Carol to deterministically reproduce the hash tree. Bob cannot add tuples not in the original collection without finding a collision in f, which is computationally infeasible. Note that, using this construction, Carol can tell how many items have been removed and from which positions in Alice's original data. This may be considered a feature and not a limitation for many applications.

The construction above is vulnerable to dictionary attacks by Carol, who can guess values for the omitted tuples and check them efficiently by hashing. In [14] a more secure signature scheme supporting controlled deletions is presented that avoids this vulnerability, along with other controlled operations including a version of deletion that does not reveal positions deleted.

The study of these extensible signature schemes is an active research area in cryptography, with a number of open problems mentioned in [24], and interesting extensibility features realized in [14, 19]. Our future goal is to adapt these techniques to our setting, as they provide an important efficiency improvement: when an extensible signature scheme exists for a useful operation, an author can effectively authorize many data elements by providing a single extensible signature. Naturally, extensible operations must be chosen carefully to avoid unintended forged signatures.

Query certification

The naive strategy for verification of a citation is to retrieve the original signed target data, compute the citation query and compare the result with the annotated data. This may be inefficient or impossible in a data exchange setting. Research into consistent query protocols [15, 22, 10, 9] can provide a more efficient verification process in some cases. These techniques allow a citation to carry proof that the derived data is the result of the citation query, relative to the summary signature on the original database. In particular, the techniques are again based on Merkle trees [17, 18] 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 $x = Q(D)$. We omit further discussion of these techniques for lack of space; please see [20] for further details.

4. MANAGING ANNOTATED DATA

Managing data that contains annotations requires representing the annotated data, expressing queries over the data, propagating annotations through queries, and interpreting the data and annotations that result.

We are motivated by data exchange scenarios, and therefore focus on semistructured data enhanced with integrity annotations. The W3C Recommendation for XML digital signature syntax [12] can provide a basis for data representation and supports the signing of arbitrary fragments of an XML document. It is easy to support multiple signatures over the same document, overlapping signatures, and signatures by multiple authors. To this basic signature schema, we wish to add metadata for extensible signatures and citations.

Querying integrity-annotated data. A query over annotated data results in output data with integrity annotations. Queries must include selection conditions over the annotations (which for instance assert that certain data elements must be signed and verified) and propagation rules which determine how signatures should be propagated from the input to the output. For example, if data in the input has multiple signatures it may be sufficient to propagate just one, or it may be necessary to carry all signatures into the output. Further, because of the flexibility of citations, a wide variety of annotations could be propagated to the output, and the choice will depend on the setting. Recent work in data provenance provides some techniques for annotation propagation. In addition users should be able to query the integrity of data declaratively, without resorting to calls to low-level cryptographic routines.

A formal model of data authenticity. A number of challenges in this area call for a formal model to analyze claims of authenticity. First, it may not be clear in all applications how an author should sign data. For instance, in Example 2.1 we considered the case where signatures of stock recommendations were applied to tickers and ratings separately, and did not secure their association. In such a simple example this flaw was immediately evident. In a more complex setting the question of what to sign – and especially what extension semantics to permit for extensible signatures – could be

a difficult issue for a data source. Second, decisions on propagation rules should be guided by the authenticity assertions that users want to verify. Finally, interpreting the meaning of multiple, possibly nested, signatures or complex citations may be very difficult. A formal model of authenticity will serve each of these issues. We have begun to address these issues by relating the authenticity guarantees of signatures to conventional database constraints [21].

5. RELATED WORK AND CONCLUSION

The authors of [4] use conventional digital signatures to implement "content extraction signatures" which allow an author to sign a document along with a definition of permissible operations of blinding (similar to redaction) and extraction. Recipients can extract data freely, but the verification procedure requires contacting the original author. The authors of [3] propose a framework of cooperative updates to a document by predetermined recipients constrained by integrity controls. In both cases, integrity properties are provided at the expense of flexible data exchange.

A number of projects [8, 6, 7] have studied data provenance, or lineage, which involves tracing and recording the origin of data and its movement among databases. These results provide important tools for managing integrity annotations, including complexity results for decision problems related to provenance and a background for propagation rules [6]. However, the emphasis of this work is not integrity, and we are concerned not just with carrying annotations, but providing cryptographic evidence of source attribution. Please see [20] for a full description of related research.

Conclusion. Today's web publishing applications require guarantees of integrity not provided by current technology. We have proposed primitives for expressing claims of origin, cryptographic techniques to implement these primitives, and have identified key problems in managing claims of integrity in the course of querying and restructuring of data. Solutions to these problems require formalizing the integrity guarantees of cryptographic primitives, and integrating these with query languages used for data management. We have tried to highlight the compelling challenge of preventing unauthorized modification of data while at the same time allowing innocuous modifications performed in the course of common collaboration and data integration.

6. REFERENCES

[1] R. Agrawal, P. J. Haas, and J. Kiernan. Watermarking relational data: framework, algorithms and analysis. *The VLDB Journal*, 12(2):157–169, 2003.

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

[3] 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.

[4] 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.

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

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

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

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

[9] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. G. Stubblebine. Flexible authentication of xml documents. In *ACM Computer and Communications Security*, pages 136–145, 2001.

[10] 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.

[11] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

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

[13] H.r. 3261, to prohibit the misappropriation of certain databases. Introduced in the House, 108th Congress, available at http://frwebgate.access.gpo.gov, 2003.

[14] R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In *RSA Conference on Topics in Cryptology*, pages 244–262, 2002.

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

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

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

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

[19] S. Micali and R. L. Rivest. Transitive signature schemes. In *RSA Conference on Topics in Cryptology*, pages 236–243. Springer-Verlag, 2002.

[20] G. Miklau. Research problems in secure data exchange. Univ. of Washington Tech Report 04-03-01, Mar 2003. Available at www.cs.washington.edu /homes/gerome.

[21] G. Miklau and D. Suciu. Modeling integrity in data exchange. In *Proceedings of VLDB 2004 Workshop on Secure Data Management*, August 2004.

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

[23] Peter Buneman and Sanjeev Khanna and Wang-Chiew Tan. Data Provenance: Some Basic Issues. In *Foundations of Software Technology and Theoretical Computer Science*, 2000.

[24] R. Rivest. Two new signature schemes. Presented at Cambridge seminar, March 2001. See http://www.cl.cam.ac.uk/Research/Security/seminars/2000/rivest-tss.pdf.

[25] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[26] Secure hash standard. *Federal Information Processing Standards Publication (FIPS PUB)*, 180(1), April 1995.

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