Certification and Authentication of Data Structures*

Roberto Tamassia[1] and Nikos Triandopoulos[1,2]

[1] Department of Computer Science, Brown University, USA.
[2] Department of Computer Science, Boston University, USA.

Abstract. We study the design of secure protocols for efficiently verifying the results of queries on data structures that are outsourced to untrusted servers, where answers are processed over relational databases in the RAM model of computation. We introduce a new authentication framework that, by design and through the new concept of query certification, aims at verifying the validity of the answer, rather than each step of the algorithm that generates the answer. Our framework efficiently reduces the authentication of general queries to that of set-membership queries, and provides sufficient conditions for achieving super-efficient answer verification (in time asymptotically less than the time spent to answer the query).

1 Introduction

The emergence of cloud computing has introduced new data management and dissemination models, where data is typically processed by untrusted, often unknown, machines. In this distributed computing setting, we would like to assure users that computations performed in the cloud have the same output as if they were locally executed.

In this paper, we study the design of secure and efficient protocols for proving the correctness of computations on data that is outsourced to remote untrusted servers. Here, a data owner outsources a database to a third-party responder who answers incoming queries on it. Assuming that the responder may *erroneously* or *maliciously* modify the query results, our goal is to augment the data structure and the associated protocols so that end-users receive efficiently verifiable proofs of the returned answers.

Verifying the output of algorithms has been the subject of research in the context of *correctness checking*, including work on the certification of programs, data structures, and graph and geometric algorithms (e.g., [6, 7, 17, 30]). However, these schemes do not fully solve our authentication problem as they protect users only against implementation errors.

Extensive research over the last decade has led to the development of *authenticated data structures* (e.g., [20]), where a data structure is augmented with the use of cryptographic primitives to store authentication information that allows to compute proofs of answers to queries. Research has initially focused on authenticated dictionaries (e.g., [3, 10, 20, 29, 31]). Various authentication schemes have been proposed for

* This work was supported in part by the U.S. National Science Foundation under grants IIS–0713403 and OCI–0724806 and by the Center for Geometric Computing at Brown University and the Center for Reliable Information Systems and Cyber Security at Boston University. The views in this paper do not necessarily reflect the views of the sponsors. We would like to thank Michael Goodrich and Charalampos Papamanthou for useful discussions.

database operations (e.g., [1, 5, 12, 15, 19, 21, 23, 25, 33]) and for geometric and graph search problems (e.g., [13, 16, 26, 27, 34]). These schemes verify the results of specific types of queries using custom constructions whose security is proved with ad hoc methods.

Preliminary progress has also been made on the design of more general authentication techniques for searching in directed acyclic graphs [13, 16] and verifying certain search algorithms [24]. However, these methods share the following characteristics: they operate in the pointer machine model of computation (i.e., queries involve searches in linked data structures); they verify step by step the computation that generates the answer (i.e., the entire search path in the data structure); and they provide solutions only for the static case (i.e., no updates are allowed in the data set). Therefore, the resulting protocols are usually less practical and possibly more costly than needed.

In this work, we propose an authentication framework that provides new techniques for systematically building *efficient and secure* data authentication schemes. We depart from previous approaches by (1) *decoupling* the answer-generation process and the answer-verification process in a *general* query model over *dynamic* data; and (2) achieving the *separation* of the algorithmic and cryptographic components in the design of data authentication schemes. Our main result is that a very general class of query types can be authenticated securely and without loss of efficiency.

We provide a formal definition for the problem of verifying query answers through *query authentication schemes* in a setting where the (honest) data owner and the (malicious) query responder are distinct entities, and where end-users do not trust (the authenticity of) the answers to their queries. Aiming at general results, we use the RAM computational model as well as a general data type and query model. The central idea in our framework is as follows: in contrast to approaches that authenticate the execution of the algorithm that answers a query, we propose an answer-based approach where only the information that is necessary for the answer verification is being authenticated.

To achieve our goal, and inspired by certifying algorithms (e.g., [7, 17]), we introduce the concept of *query certification*, which models answer verification in the client-server model and in an information-theoretic sense (i.e., the server cannot cheat the client). In particular, a certification data structure for a query type defines information and corresponding algorithms that are sufficient to verify the correctness of the answer to any concrete query. We then identify the inherent relationship between query certification and query authentication, and put forward a new approach for data authentication. We show that for any query type, we can build an authenticated data structure that provides authenticated queries in the bounded computational model (i.e., under some hardness assumptions, the responder cannot cheat a user) by first designing a certification data structure for the same query type and then applying simple cryptographic constructions to its functionality. Our framework thus achieves modularity by separating the algorithmic from the cryptographic design considerations: to build an authenticated data structure, one can first simply design a related certification data structure, which, in turn, is automatically transformed into a query authentication scheme.

Moreover, this transformation of a certification data structure into an authenticated data structure satisfies an important property: to verify an answer to a general query, the authenticated data structure can only use a query authentication scheme for *set-*

membership queries, that is, protocols that verify membership in sets (actually, only positive answers to these queries). To show this property, we introduce the concept of query reducibility in data authentication. Informally, we say that query of type A is *authentically reduced* to query of type B if a query authentication scheme for B leads to a query authentication scheme for A. We thus show that any query problem over relational databases is authentically reduced to the fundamental set-membership problem in a way that preserves efficiency. The usefulness of this reduction stems from the fact that several efficient cryptographic constructions exist for set-membership query authentication schemes, including Merkle trees [18], distributed Merkle trees [9, 32], one-way accumulators [4, 22], and authenticated skip lists [8, 11, 28, 31] and hash tables [29]).

Finally, our framework yields a method for achieving *super-efficient* data authentication, where verifying the answer is asymptotically faster than answering the query. Indeed, although the above completeness result is proven by verifying the query-answering algorithm (thus extending previous possibility results [16] to general queries over dynamic data), we demonstrate that super-efficient certification structures exist for certain query types. This allows us to exploit the computational gap that is often observed between answering a query and verifying its answer (see, e.g., [12]), and to leverage existing results on correctness checking in data authentication. Overall, super-efficient verification can be achieved in our framework by designing new super-efficient certification structures, or by constructing new cryptographic primitives for optimal set-membership verification, or by improving on both directions.

The rest of this paper is organized as follows. In Section 2, we introduce certification data structures, which model answer testability, a core concept in our work, and provide a constructive proof of their existence. In Section 3, we describe query authentication schemes. In Section 4, we present our main possibility results for data authentication and, in Section 5, we present some applications of our framework.

2 Certification Data Structures

A (structured) *data set* $S = (\mathcal{E}, \mathcal{R})$ consists of: (i) a collection $\mathcal{E} = \{E_1, \ldots, E_t\}$ of sets of data elements such that, for $1 \leq i \leq t$, set E_i is a subset of a *universe* $\mathcal{U}_i$, and (ii) a collection $\mathcal{R} = \{R_1, \ldots, R_k\}$ of indexed sequences of tuples of data elements such that, for $1 \leq i \leq k$, sequence $R_i = (R_i[1], \ldots, R_i[m_i])$ consists of m_i distinct p_i-tuples from $E_{j_1} \times \ldots \times E_{j_{p_i}}$, where $1 \leq j_1 \leq \ldots \leq j_{p_i} \leq t$ and $p_i < p$ for some integers p and m_i. The *size* n of data set $S = (\mathcal{E}, \mathcal{R})$ is defined as $n = \sum_{i=1}^{t} |E_i|$. Also, we assume that t, k and p are constants (with respect to n).

Our data model shares concepts from the relational database model using indexed sequences of tuples, i.e., each member R_i of $\mathcal{R}$ is an array of tuples, where each tuple can be indexed by an integer. Thus, very general data organization and algorithmic paradigms are captured. For instance, a graph $G = (V, E)$ may correspond to data set $S_G = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E} = V$ and $\mathcal{R}$ consists of a single sequence of indexed pairs representing relation E (edges in G). More complex graphs, e.g., with edge directions, weights, costs or associated data elements, can be represented by appropriately including new primitive data-element sets in $\mathcal{E}$ and corresponding sequences in $\mathcal{R}$ describing the structure of data elements as well as various relations among them.

A *query operation* Q_S on data set $S = (\mathcal{E}, \mathcal{R})$ is a computable function $Q_S : \mathcal{Q} \rightarrow \mathcal{A}_S$, where $\mathcal{Q}$ is the query space (the set of all possible queries q of specific type that can be issued about S) and $\mathcal{A}_S$ is the answer space (the set of all possible answers to queries on S drawn from $\mathcal{Q}$). The *answer* of a query $q \in \mathcal{Q}$ under Q_S is $Q_S(q) \in \mathcal{A}_S$. An element $a \in \mathcal{A}_S$ of the answer space is the *correct answer* for query q if and only if $Q_S(q) = a$. The above description captures general query operations on data sets that are based on relations. The only requirement is that any query in the query space is mapped to a unique answer in the answer space and that any answer corresponds to some query.[1]

For example, if $S_G = (\mathcal{E}, \mathcal{R})$ represents a subdivision of the plane into the polygons induced by the vertices and edges of a planar graph G embedded in the plane, the *point location* query operation maps a point in the plane (query) to the unique element of the subdivision (region, edge, or vertex) containing it (answer).

Regarding the complexity of query answering, we only require that query operation Q_S is efficiently computable. Typically, function Q_S is evaluated on query $q \in \mathcal{Q}$ by an algorithm that operates over S through a data structure that supports queries in $\mathcal{Q}$. A *query data structure* $D(Q_S)$ for a query operation $Q_S : \mathcal{Q} \rightarrow \mathcal{A}_S$ on data set $S = (\mathcal{E}, \mathcal{R})$ consists of a structured data set $(\mathcal{E}_Q, \mathcal{R}_Q)$, such that $\mathcal{E} \subset \mathcal{E}_Q$ and $\mathcal{R} \subset \mathcal{R}_Q$ and an algorithm Answer, which on input a query $q \in \mathcal{Q}$ and data set $(\mathcal{E}_Q, \mathcal{R}_Q)$, returns $Q_S(q) \in \mathcal{A}_S$ in time polynomial in n and $|q|$ by accessing and processing tuples in $\mathcal{R}$. We write $D(Q_S) = (\mathcal{E}_Q, \mathcal{R}_Q, \mathsf{Answer})$. Given input query q, algorithm Answer operates over S through the use of $D(Q_S)$: by processing relations in $\mathcal{R}_Q$, Answer accesses relations in $\mathcal{R}$, evaluates conditions over elements in S and produces the answer.

For example, for the point location algorithm based on the chain method [14], data set $(\mathcal{E}_Q, \mathcal{R}_Q)$ represents a two-level search structure; here, data set $S_G = (\mathcal{E}, \mathcal{R})$ includes information about the regions defined by the embedded planar graph G.

A data set S is *static* if it stays the same over time and *dynamic* if it evolves through *update operations* performed on S. An update operation U_S for S is a function that given an update $y \in \mathcal{Y}$, $\mathcal{Y}$ being the set of all possible updates, results in changing one or more data elements in $\mathcal{E}$ and one or more tuples in $\mathcal{R}$. If S is static (resp. dynamic), data set $(\mathcal{E}_Q, \mathcal{R}_Q)$ can be constructed (resp. updated) by some algorithm Constr_Q (resp. Update_Q) that runs on input S (resp. S and $y \in \mathcal{Y}$) in polynomial time in n.

Our data querying model achieves generality by combining the expressiveness of relational databases with the power of the RAM computation model. By using index-annotated relations, complex data organizations are easily represented and accessed. For instance, indirect addressing is supported by treating indexes as a distinct data type which is included in $\mathcal{E}$. Thus, our model strictly contains the pointer machine model.

We now explore the decoupling of query answering and answer verification. We start by defining the notion of *answer testability*, formally expressed through a *certification data structure*. This notion captures the following property in data querying: query operations on any data set return verifiable answers that can be tested to be cor-

[1] Unique answers are used without loss of generality as we can appropriately augment the query space to include the index of the answer (according a fixed ordering) that we wish to obtain.

rect given a (minimal) subset of specially selected relations over elements of the data set. In essence, queries are information-theoretically certified to return valid answers.

Definition 1 (Certification Data Structure). *Let $D(Q_S) = (\mathcal{E}_Q, \mathcal{R}_Q, \mathsf{Answer})$ be a query data structure for query operation $Q_S : \mathcal{Q} \to \mathcal{A}_S$ on data set $S = (\mathcal{E}, \mathcal{R})$ of size n. A* certification data structure *for S with respect to $D(Q_S)$ is a triplet $C(Q_S) = ((\mathcal{E}_C, \mathcal{R}_C), \mathsf{Certify}, \mathsf{Verify})$, where $(\mathcal{E}_C, \mathcal{R}_C)$, called the* certification image *of S, is a structured data set and* Certify *and* Verify *are algorithms with the following properties:*

Answer tests: *On input query $q \in \mathcal{Q}$ and data sets $(\mathcal{E}_Q, \mathcal{R}_Q)$ and $(\mathcal{E}_C, \mathcal{R}_C)$, algorithm* Certify *returns answer $a = Q_S(q)$ and an* answer test *τ, which is a sequence of pairs (i, j), each indexing a tuple $R_i[j]$ of $\mathcal{R}_C$. Answer test τ defines a subset $\mathcal{R}_C(\tau) \subseteq \mathcal{R}_C$, called the* certification support *of answer a.*

Answer testability: *On input query $q \in \mathcal{Q}$, data set $(\mathcal{E}_C, \mathcal{R}_C)$, answer $a \in \mathcal{A}_S$ and answer test τ, algorithm* Verify *accesses and processes* only *relations in $\mathcal{R}_C(\tau)$ and returns either 0 (rejects) or 1 (accepts).*

Test reliability: *(i) For all $q \in \mathcal{Q}$,* $\mathsf{Verify}(q, \mathcal{R}_C, \mathsf{Certify}(q, (\mathcal{E}_Q, \mathcal{R}_Q), (\mathcal{E}_C, \mathcal{R}_C))) = 1$ *(*completeness*); and (ii) for all queries q, answers a, and answer tests τ, whenever it is* $\mathsf{Verify}(q, \mathcal{R}_C, a, \tau) = 1$*, we have that $a = Q_S(q)$ (*soundness*).*

Regarding complexity measures for certification data structure $C(Q_S)$, we say: (1) $C(Q_S)$ is answer-efficient *if the time complexity $T_C(n)$ of* Certify *is asymptotically at most the time complexity $T_A(n)$ of* Answer*, i.e., $T_C(n)$ is $O(T_A(n))$; (2) $C(Q_S)$ is* time-efficient *(resp.* time super-efficient*) if the time complexity $T_V(n)$ of* Verify *is asymptotically at most (resp. less than) the time complexity $T_A(n)$ of* Answer*, i.e., $T_V(n)$ is $O(T_A(n))$ (resp. $o(T_A(n))$); and analogously (3) $C(Q_S)$ is* space-efficient *(resp. space super-efficient) if the space requirement $S_C(n)$ of $(\mathcal{E}_C, \mathcal{R}_C)$ is asymptotically at most (resp. less than) the space requirement $S_Q(n)$ of $(\mathcal{E}_Q, \mathcal{R}_Q)$, i.e., $S_C(n)$ is $O(S_Q(n))$ (resp. $o(S_Q(n))$). If S is static, data set $(\mathcal{E}_C, \mathcal{R}_C)$ can be constructed by some algorithm Constr_C that runs on input S in polynomial time in n.*

For simplicity, the above definition corresponds to the static case. The dynamic case can be defined similarly. Informally, an update algorithm Update_C is responsible to handle updates in data set S by accordingly updating $C(Q_S)$; that is, it produces the updated set $(\mathcal{E}'_C, \mathcal{R}'_C)$ and, in particular, the set of tuples where $\mathcal{R}'_C$ and $\mathcal{R}_C$ differ. Algorithm Update_C additionally produces an *update test* (as the answer test above, a set of indices for tuples in $\mathcal{R}_C$) that validates the performed changes. Similarly, an update testing algorithm Updtest, on input an update $y \in \mathcal{Y}$, set $\mathcal{R}_C$, a set of tuples (changes in $\mathcal{R}_C$) and an update test, accepts if and only if the tuples correspond to the correct, according to y, new or deleted tuples in $\mathcal{R}_C$. Similarly, we can define *update efficiency* and *update-testing (super-)efficiency* for $C(Q_S)$, with respect to the time complexity of Update_C and Updtest, respectively, as they asymptotically compare to Update_Q.

Certification data structures introduce a new dimension in the study of data querying and answer validation. They support certification of queries in a setting where the notions of query answering and answer validation are conceptually and algorithmically separated in a clean way. In particular, answer validation is based merely on the certification image $(\mathcal{E}_C, \mathcal{R}_C)$ of $S = (\mathcal{E}, \mathcal{R})$ (these two sets share tuples, possibly, through a

subset relation) and not on set $(\mathcal{E}_Q, \mathcal{R}_Q)$ of the query data structure. Also, query certification depends *only* on the certification support of the answer, i.e., subset $\mathcal{R}(\tau)$.

We first prove that for every query type, there is an efficient certification structure, a completeness result showing that all queries can be certified without loss of efficiency.

Theorem 1. *Any query data structure for any query operation on a data set admits a certification data structure that is answer-efficient, time-efficient, update-testing-efficient, and space-efficient.*

Proof. (Sketch.) We discuss the static case; the dynamic case is treated analogously. Let $S = (\mathcal{E}, \mathcal{R})$ be a data set of size n, Q_S a query operation on S and $D(Q_S) = (\mathcal{E}_Q, \mathcal{R}_Q, \mathsf{Answer})$ a query data structure for Q_S. We describe a certification data structure $C(Q_S) = ((\mathcal{E}_C, \mathcal{R}_C), \mathsf{Certify}, \mathsf{Verify})$ for S with respect to $D(Q_S)$. First we set $(\mathcal{E}_C, \mathcal{R}_C) = (\mathcal{E}_Q, \mathcal{R}_Q)$. Algorithm Certify is an augmented version of Answer. Given a query $q \in \mathcal{Q}$ and sets $(\mathcal{E}_C, \mathcal{R}_C)$, $(\mathcal{E}_Q, \mathcal{R}_Q)$, Certify creates an empty sequence τ of indices of tuples in $\mathcal{R}_C$ and then it runs Answer on input $(q, (\mathcal{E}_Q, \mathcal{R}_Q))$ to produce the answer $Q_S(q)$. Also, any time algorithm Answer accesses a tuple $R_i[j]$ in $\mathcal{R}_Q$, algorithm Certify adds (i, j) to the end of sequence τ. When Answer terminates, so does Certify, and returns the output $a = Q_S(q)$ produced by Answer and sequence τ as the corresponding answer test. Algorithm Verify as an augmented version of Answer operating as follows. On input a query $q \in \mathcal{Q}$, set $(\mathcal{E}_C, \mathcal{R}_C)$, an answer a and a sequence τ, algorithm Verify starts executing algorithm Answer on input $(q, (\mathcal{E}_Q, \mathcal{R}_Q))$ and checks the execution of Answer subject to sequence τ. That is, each time Answer retrieves a tuple $R_i[j]$ in $(\mathcal{E}_Q, \mathcal{R}_Q)$, Verify removes the first element of τ and compares it to (i, j), rejecting the input if the comparison fails. When Answer terminates, the answer computed by Answer is compared with the answer provided as input: if the two answers agree (are equal) then Verify accepts its input, otherwise it rejects.

Completeness and soundness conditions are satisfied by construction. Finally, our certification data structure is answer-, time- and space-efficient. This follows from the fact that for any inputs, Certify and Verify do a total amount of work that is only by a constant factor more than the work of Answer, thus $T_C(n) = O(T_A(n))$ and $T_V(n) = O(T_A(n))$, and the fact that $(\mathcal{E}_C, \mathcal{R}_C) = (\mathcal{E}_Q, \mathcal{R}_Q)$, thus $S_C(n) = O(S_Q(n))$. Observe that each pair (i, j) in the answer test τ is accessed in constant time. □

Certification data structures are designed to accompany two-party data query protocols as follows: party A possesses sets $(\mathcal{E}_Q, \mathcal{R}_Q)$ and $(\mathcal{E}_C, \mathcal{R}_C)$ and runs Certify, and party B possesses set $(\mathcal{E}_C, \mathcal{R}_C)$ and runs Verify. The underlying outsourced set S is controlled by B by creating update and query operations for S. Although both operations are performed at A, B is able to verify their correctness. Thus, this setting models *certified outsourced computation*: at any point in time, B maintains a correct certification image of S, allowing verification *without* loss of efficiency, by Theorem 1. Although its existential proof is simple (both parties execute the same algorithms on the same data) yet, its significance is justified by the following:

1. In addition to showing that Definition 1 is meaningful, Theorem 1 proves the *feasibility of answer testability for any computable query* in a general querying and computational model.

2. *Time super-efficient certification is in general feasible* (e.g., for range searching [12], related SQL queries, point location, convex hull [17]), so outsourced computations remain meaningful. For instance, super-efficient certification for point location can be achieved using a trapezoidal decomposition of the subdivision as the certification image. In this setting, the answer test for a point location corresponds to a trapezoid containing the point.
3. Using cryptography in the bounded computational model, data certification can be used to achieve *authentication* and *consistency* in third-party models and *space super-efficiency* of certified outsourced computations in client-server models (see Section 5).

3 Authenticated Data Structures

We describe a general model for data authentication by introducing *query authentication schemes*, cryptographic protocols for verifying the results of general queries over data sets. We extend certification structures to achieve correctness validation (in a computational sense) in a setting where the owner of a data set does not control the data structure used to answer queries. Instead, queries are answered by an untrusted, possibly malicious, party, and answers are augmented with a proof used to verify validity.

A *three-party data querying model* consists of a *source* $\mathcal{S}$, a *responder* $\mathcal{R}$ and a *user* $\mathcal{U}$, where: (1) source $\mathcal{S}$ creates (and owns) a dynamic data set S, which is maintained by query data structure $D(Q_S)$ for query operation $Q_S : \mathcal{Q} \to \mathcal{A}_S$ on S; (2) responder $\mathcal{R}$ stores S, by maintaining a copy of $D(Q_S)$ and some auxiliary information for S; (3) user $\mathcal{U}$ issues queries about S to responder $\mathcal{R}$ by sending to $\mathcal{R}$ a query $q \in \mathcal{Q}$; (4) on a query $q \in \mathcal{Q}$ issued by $\mathcal{U}$, $\mathcal{R}$ computes answer $a = Q_S(q)$ and sends a to $\mathcal{U}$; (5) on an update $y \in \mathcal{Y}$ for S issued by $\mathcal{S}$, S and $D(Q_S)$ are appropriately updated by $\mathcal{S}$ and $\mathcal{R}$.

Definition 2 (Query Authentication Scheme). *A* query authentication scheme *for query operation $Q_S : \mathcal{Q} \to \mathcal{A}_S$ on set S is a quadruple of algorithms* $(\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$ *s.t.:*

***Key generation*:** *Algorithm* G *takes as input a security parameter* 1^κ*, and outputs a key pair* (PK, SK)*. We write* $(PK, SK) \leftarrow \mathsf{G}(1^\kappa)$.

***Authenticator*:** *Algorithm* Auth *takes as input the secret and public key* (SK, PK)*, the query space* $\mathcal{Q}$ *(or an encoding of the query type) and data set* S *of size* n *and outputs an authentication string* α *and a verification structure* V*, that is* $(\alpha, \mathsf{V}) \leftarrow \mathsf{Auth}(SK, PK, \mathcal{Q}, S)$*, where* $\alpha, \mathsf{V} \in \{0,1\}^*$.

***Responder*:** *Algorithm* Res *takes as input a query* $q \in \mathcal{Q}$*, a data set* S *of size* n *and a verification structure* $\mathsf{V} \in \{0,1\}^*$ *and outputs an answer-proof pair* $(a, p) \leftarrow \mathsf{Res}(q, S, \mathsf{V})$*, where* $a \in \mathcal{A}_S$ *and* $p \in \{0,1\}^*$.

***Verifier*:** *Algorithm* Ver *takes as input the public key* PK*, a query* $q \in \mathcal{Q}$*, an answer-proof pair* $(a, p) \in \mathcal{A}_S \times \{0,1\}^*$ *and an authentication string* $\alpha \in \{0,1\}^*$ *and either accepts the input, or rejects, i.e.,* $\{0,1\} \leftarrow \mathsf{Ver}(PK, q, (a, p), \alpha)$.

***Updates*:** *For the dynamic case, we additionally require the existence of an update algorithm* Auth_U *that complements algorithm* Auth *and handles updates; namely,* Auth_U *given update* $y \in \mathcal{Y}$*, it updates the authentication string and the verification structure:* $(\alpha', \mathsf{V}') \leftarrow \mathsf{Auth}_U(SK, PK, \mathcal{Q}, S, y, \alpha, \mathsf{V})$.

We require two properties for a query authentication scheme: *correctness* and *security*. Correctness mandates that a verification algorithm should accept valid answer-proof pairs generated by the responder. Security dictates that, given any query issued by $\mathcal{U}$, a computationally bounded malicious responder $\mathcal{R}$ cannot compute an incorrect answer and an associated proof such that $\mathcal{U}$ accepts this pair (thus, incorrectly validating a fake answer). In our three-party data querying model, we assume that $\mathcal{U}$ trusts $\mathcal{S}$ but not $\mathcal{R}$, and that $\mathcal{R}$ always participates in the protocol (i.e., we do not consider denial-of-service attacks).

Definition 3 (Correctness & Security). *We say that a query authentication scheme* $(\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$ *is* correct *if for all queries* $q \in \mathcal{Q}$*, if* $(\alpha, \mathsf{V}) \leftarrow \mathsf{Auth}(SK, PK, \mathcal{Q}, S)$ *and additionally* $(a, p) \leftarrow \mathsf{Res}(q, S, \mathsf{V})$*, then with overwhelming probability it holds that* $1 \leftarrow \mathsf{Ver}(PK, q, (a, p), \alpha)$ *and* $Q_S(q) = a$*. We say that a query authentication scheme* $(\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$ *for query operation* $Q_S : \mathcal{Q} \rightarrow \mathcal{A}_S$ *on structured data set* S *is* secure*, if no probabilistic polynomial-time adversary* $\mathcal{A}$*, given any query* $q \in \mathcal{Q}$*, public key* PK *and oracle access to the authenticator algorithm* Auth*, can output an authentication string* α*, an answer* a' *and a proof* p'*, such that* a' *is an incorrect answer that passes the verification test, that is,* $a' \neq Q_S(q)$ *and* $1 \leftarrow \mathsf{Ver}(PK, q, (a', p'), \alpha)$*.*

We say that a *correct* and *secure* query authentication scheme $(\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$ for queries in query space $\mathcal{Q}$ on a data set S constitutes an *authenticated data structure* for this type of queries. It is implied that, given an authentication string α, for algorithm Ver it holds that, for all queries $q \in \mathcal{Q}$, with all but negligible probability (that is measured over the probability space of the responder algorithm) it holds that $Q_S(q) = a$ if and only if there exists p s.t. $1 \leftarrow \mathsf{Ver}(PK, q, (a, p), \alpha)$.

4 Authentication Reductions

We now describe the main results of our work. First, we introduce the notion of reducibility in data authentication by defining reductions between query authentication schemes. We then prove that the authentication of any query is reduced to the authentication of *set membership* queries. In fact, we need to authenticate only positive answers—that is, relation $\in$, but not $\notin$, is authenticated. Next, we present implications of this result.

Using certification structures, we provide a general methodology for constructing correct and secure query authentication schemes. Also, based on super-efficient query certification, we develop a method for data authentication where only the information necessary for answer verification is authenticated, and not the entire execution of the search algorithm, which leads to a framework for the design of authentication structures with super-efficient verification.

Let $QAS(Q_S, S)$ denote a query authentication scheme (QAS) for Q_S and S. Authenticated reductions among QASs allow the design of a new QAS using no other tools but only what another QAS provides, in a way that preserves correctness and security.

In particular, let S and S' be data sets, $Q_S : \mathcal{Q} \rightarrow \mathcal{A}_S$, $Q'_S : \mathcal{Q}' \rightarrow \mathcal{A}'_S$ be query operations on S and S' respectively, and $QAS(Q_S, S) = (\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$, $QAS(Q'_S, S') = (\mathsf{G}', \mathsf{Auth}', \mathsf{Res}', \mathsf{Ver}')$ be query authentication schemes for Q_S on

S and Q'_S on S' respectively. We say that $QAS(Q_S, S)$ is *authentically reduced* to $QAS(Q'_S, S')$, if key generation algorithms G and G' are identical, $QAS(Q_S, S)$ uses the public and secret keys generated by G *never explicitly*, but *only implicitly* through black-box invocations of algorithms Auth', Res' and Ver', and $QAS(Q_S, S)$ is correct and secure whenever $QAS(Q'_S, S')$ is correct and secure.

Let $S = (\mathcal{E}, \mathcal{R})$ be a structured data set and $Q_S : \mathcal{Q} \to \mathcal{A}_S$ be any query operation. Let $D(Q_S) = (\mathcal{E}_Q, \mathcal{R}_Q, \mathsf{Answer})$ be a query data structure for Q_S. By Theorem 1, we know that there exists a certification data structure $C(Q_S) = ((\mathcal{E}_C, \mathcal{R}_C), \mathsf{Certify}, \mathsf{Verify})$ for S with respect to Q_S. Let $Q_\in : \mathcal{Q}(\mathcal{R}_C) \to \{\mathsf{yes}, \mathsf{no}\}$ be the set membership query operation, where the query space $\mathcal{Q}(\mathcal{R}_C)$ is the indexed tuples that exist in $\mathcal{R}_C$. Assuming the existence of a secure and correct $QAS(Q_\in, \mathcal{R}_C) = (\mathsf{G}', \mathsf{Auth}', \mathsf{Res}', \mathsf{Ver}')$, we construct $QAS(Q_S, S) = (\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$, a query authentication scheme for Q_S and S, parameterized by $QAS(Q_\in, \mathcal{R}_C)$ for set membership queries.

Key-generation algorithm. Same as G', thus, $SK = SK'$ and $PK = PK'$.

Authenticator. The authenticator algorithm Auth computes structured data set $S_C = (\mathcal{E}_C, \mathcal{R}_C)$ of the certification structure $C(Q_S) = ((\mathcal{E}_C, \mathcal{R}_C), \mathsf{Certify}, \mathsf{Verify})$ using S and Constr_C. Then Auth runs algorithm Auth' on input SK', PK', $Q_\in$ and $\mathcal{R}_C$. Then, Auth outputs the pair $(\alpha', \mathsf{V}') \leftarrow \mathsf{Auth}'(SK', PK', Q_\in, \mathcal{R}_C)$.

Responder. The responder algorithm Res first computes the structured data sets $S_Q = (\mathcal{E}_Q, \mathcal{R}_Q)$ and $S_C = (\mathcal{E}_C, \mathcal{R}_C)$ using S and algorithms Constr_Q and Constr_C. Then, on input q, S_Q and S_C it simply runs algorithm $\mathsf{Certify}$ to produce its pair (a, τ). Then Res constructs the certification support $\mathcal{R}_C(\tau)$ of answer a by accessing set $\mathcal{R}_C$ with the use of indices in τ. For every tuple $\langle t \rangle$ in $\mathcal{R}_C$, algorithm Res runs the responder algorithm Res' on inputs $\langle t \rangle$, $\mathcal{R}_C$ and V' to get $(a'(t), p'(t)) \leftarrow \mathsf{Res}'(\langle t \rangle, \mathcal{R}_C, \mathsf{V}')$ and, if $(t_1, \ldots, t_{|\tau|})$ is the sequence of tuples accessed in total, Res creates sequence $p' = (p'(t_1), \ldots, p'(t_{|\tau|}))$, sets $p = (\tau, \mathcal{R}_C(\tau), p')$ and finally outputs (a, p).

Verifier. The verifier algorithm Ver first checks if the proof p and answer a are both well-formed and, if not, it rejects. Otherwise, by appropriately processing the proof p, algorithm Ver runs algorithm Verify on inputs q, $\mathcal{R}_C(\tau)$, a and τ. Whenever algorithm Verify needs to access and process a tuple $\langle t_i \rangle$, where $\langle t_i \rangle$ is the i-th tuple accessed by Verify, algorithm Ver runs Ver' on inputs PK', $\langle t_i \rangle$, $(\mathsf{yes}, p'(t_i))$ and α' and if $0 \leftarrow \mathsf{Ver}'(PK', \langle t_i \rangle, (\mathsf{yes}, p'(t_i)), \alpha')$, algorithm Ver rejects. Otherwise, Ver continues with the computation. Finally, Ver accepts if and only if Verify accepts.

We have thus constructed $QAS(Q_S, S)$, where Q_S is a general query operation of set S, parameterized by $QAS(Q_\in, \mathcal{R}_C)$, where $Q_\in$ is the set membership query and $\mathcal{R}_C$ is the certification image of S with respect to the certification data structure in use.

Theorem 2. *Let $QAS(Q_\in, \mathcal{R}_C)$ be a correct and secure query authentication scheme for set membership queries. Given any data set S and query operation Q_S, the query authentication scheme $QAS(Q_S, S)$ constructed above is correct and secure.*

Proof. (Sketch.) We start by first discussing the correctness property. Suppose that query authentication scheme $(\mathsf{G}', \mathsf{Auth}', \mathsf{Res}', \mathsf{Ver}')$ is correct. We want to show that scheme $(\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$ is correct. This follows from checking that the verifier Ver

does not reject when given an answer-proof pair from the responder Res, for any query issued in $\mathcal{Q}$. Indeed, from the completeness property of the certification data structure the answer testing algorithm Verify does not reject, and additionally the correctness of $(\mathsf{G}', \mathsf{Auth}', \mathsf{Res}', \mathsf{Ver}')$ guarantee that Ver does not reject because of a rejection by Ver'. For the security we argue as follows. Suppose that $(\mathsf{G}', \mathsf{Auth}', \mathsf{Res}', \mathsf{Ver}')$ is secure. Assume that $(\mathsf{G}, \mathsf{Auth}, \mathsf{Res}, \mathsf{Ver})$ is not secure, then with non-negligible probability Res responds to a query $q \in \mathcal{Q}$ incorrectly but still Ver fails to reject its input. Based on the soundness property of the certification data structure in use, we must admit that it is not Certify that cheats the verifier, that is, it is not the indices in sequence τ that cause the problem, but rather the fact that algorithm Verify runs on incorrect data. Then there must exist at least one element verified to be a member of $\mathcal{R}_C$ that it is not authentic, meaning that its index is correct but one or more of the data elements in the tuple have been (maliciously) altered. We thus conclude that for at least one query the verification algorithm Ver' of query authentication scheme $(\mathsf{G}', \mathsf{Auth}', \mathsf{Res}', \mathsf{Ver}')$ failed to reject on an invalid query-answer pair, a contradiction. □

Theorem 3. *For any query operation Q_S on any data set S, there exists a secure and correct query authentication scheme $QAS(Q_S, S)$ based on a certification data structure $C(Q_S)$.*

Proof. It follows from Theorem 1, Theorem 2 and the fact that there exist secure query authentication schemes for membership queries. □

By Theorem 3, all query operations can be authenticated in the three-party authentication model given a corresponding certification structure, which can be constructed for query problems from the query data structure (Theorem 1). Our results hold for the RAM model of computation, which strictly includes the pointer machine model. Our framework can be further extended to super-efficient verification, as described in the following theorem.

Theorem 4. *Let S be a data set and $\mathcal{Q}_s$ be a query operation on S. If there exists a time (space) super-efficient certification data structure for Q_S, then there exists a time (space) super-efficient authenticated data structure for Q_S.*

5 Applications

We briefly overview applications of our framework. In the bounded computational model, we can achieve *storage outsourcing*, where the certification image $(\mathcal{E}_C, \mathcal{R}_C)$ of a data set S is stored by an untrusted entity. Consider a certification data structure, where party A (server computations are outsourced to) runs Certify and party B (client data originates from) runs Verify. It is possible for B to store only a cryptographic commitment of $(\mathcal{E}_C, \mathcal{R}_C)$ and still be able to verify its integrity throughout a series of updates on an initially empty set S. We refer to this property as *consistency*: data source B checks that any update on S, and thus on $(\mathcal{E}_C, \mathcal{R}_C)$, is in accordance with the history of previous updates. The idea is to use cryptographic primitives (e.g., hashing) that provide commitments of sets, subject to which membership can be securely checked. The

following result finds applications to cloud storage, where a client (small computational device, B) uses space super-efficient protocols to check the integrity of data stored at an untrusted server (A).

Theorem 5. *In the bounded computational model, any certification data structure can be transformed to a consistent client-server data outsourcing scheme with constant-space storage at the client.*

This result can be used to extend the construction of Section 4, where the data source stores the certification image, while the responder stores the data set and the certification image (parties B and A in our previous discussion). Using Theorem 5, the data source can outsource the certification image to the responder and still be able to check the data correctness and consistency, using constant space (similar to [2]).

Theorem 6. *For any query operation Q_S on any data set S, there exists a secure and correct query authentication scheme $QAS(Q_S, S)$ based on a certification data structure $C(Q_S)$ such that the source uses constant-space storage.*

Finally, applying our results to the peer-to-peer Merkle tree realization [9, 32] (a distributed authenticated dictionary that provides a secure *distributed* query authentication scheme for membership queries), we obtain that all queries can be authenticated with a responder implemented over a distributed peer-to-peer network.

Theorem 7. *For any query operation on data sets, there exists a query authentication scheme with a distributed responder and with constant-space storage used at the source.*

References

[1] M. J. Atallah, Y. Cho, and A. Kundu. Efficient data authentication in an environment of untrusted third-party distributors. *Proc. ICDE*, pp. 696–704, 2008.

[2] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[3] A. Buldas, P. Laud, and H. Lipmaa. Accountable certificate management using undeniable attestations. *Proc. CCS*, pp. 9–17, 2000.

[4] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. *Proc. CRYPTO*, pp. 61–76, 2002.

[5] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic data publication over the Internet. *Journal of Computer Security*, 11(3):291 – 314, 2003.

[6] G. Di Battista and G. Liotta. Upward planarity checking: "Faces are more than polygons". *Proc. GD*, pp. 72–86, 1998.

[7] U. Finkler and K. Mehlhorn. Checking priority queues. *Proc. SODA*, pp. 901–902, 1999.

[8] M. T. Goodrich, C. Papamanthou, and R. Tamassia. On the cost of persistence and authentication in skip lists. *Proc. WEA*, pp. 94–107, 2007.

[9] M. T. Goodrich, J. Z. Sun, R. Tamassia, and N. Triandopoulos. Reliable resource searching in peer-to-peer networks. *Proc. SecureComm*. pp. 437–447, 2009.

[10] M. T. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. *Proc. ISC*, pp. 372–388, 2002.

[11] M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. *Proc. DISCEX II*, pp. 68–82, 2001.

[12] M. T. Goodrich, R. Tamassia, and N. Triandopoulos. Super-efficient verification of dynamic outsourced databases. *Proc. CT–RSA*, pp. 407–424, 2008.
[13] M. T. Goodrich, R. Tamassia, N. Triandopoulos. Authenticated data structures for graph and geometric searching. *Algorithmica*, DOI 10.1007/s00453-009-9355-7, 2010.
[14] D. T. Lee and F. P. Preparata. Location of a point in a planar subdivision and its applications. *SIAM J. Comput.*, 6(3):594–606, 1977.
[15] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. *Proc. SIGMOD*, pp. 121–132, 2006.
[16] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1):21–41, 2004.
[17] K. Mehlhorn, S. Näher, M. Seel, R. Seidel, T. Schilz, S. Schirra, and C. Uhrig. Checking geometric programs or verification of geometric structures. *CGTA*, 12(1–2):85–103, 1999.
[18] R. C. Merkle. A certified digital signature. *Proc. CRYPTO*, pp. 218–238, 1989.
[19] G. Miklau and D. Suciu. Implementing a tamper-evident database system. *Proc. ASIAN*, pp. 28–48. 2005.
[20] M. Naor and K. Nissim. Certificate revocation and certificate update. *Proc. USENIX Security*, pp. 217–228, 1998.
[21] M. Narasimha and G. Tsudik. Authentication of outsourced databases using signature aggregation and chaining. *Proc. DASFAA*, pp. 420–436, 2006.
[22] L. Nguyen. Accumulators from bilinear pairings and applications. *Proc. CT-RSA* pp. 275–292, 2005.
[23] G. Nuckolls. Verified query results from hybrid authentication trees. *Proc. DBSec*, pp. 84–98, 2005.
[24] R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs for generalized queries on a committed database. *Proc. ICALP*, pp. 1041–1053, 2004.
[25] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying completeness of relational query results in data publishing. *Proc. SIGMOD*, pp. 407–418, 2005.
[26] H. Pang and K. Mouratidis. Authenticating the query results of text search engines. *PVLDB*, 1(1):126–137, 2008.
[27] S. Papadopoulos, D. Papadias, W. Cheng, and K.-L. Tan. Separating authentication from query execution in outsourced databases. *Proc. ICDE*, pp. 1148–1151, 2009.
[28] C. Papamanthou and R. Tamassia. Time and space efficient algorithms for two-party authenticated data structures. *Proc. ICICS*, pp. 1–15, 2007.
[29] C. Papamanthou, R. Tamassia, and N. Triandopoulos. Authenticated hash tables. *Proc. CCS*, pp. 437–448, 2008.
[30] G. F. Sullivan, D. S. Wilson, and G. M. Masson. Certification of computational results. *IEEE Trans. Comput.*, 44(7):833–847, 1995.
[31] R. Tamassia and N. Triandopoulos. Computational bounds on hierarchical data processing with applications to information security. *Proc. ICALP*, pp. 153–165, 2005.
[32] R. Tamassia and N. Triandopoulos. Efficient content authentication in peer-to-peer networks. *Proc. ACNS*, pp. 354–372, 2007.
[33] Y. Yang, D. Papadias, S. Papadopoulos, and P. Kalnis. Authenticated join processing in outsourced databases. *Proc. SIGMOD*, pp. 5–18, 2009.
[34] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Authenticated indexing for outsourced spatial databases. *VLDB J.*, 18(3):631–648, 2009.