Secure Authenticated Comparisons*

Keith B. Frikken[1], Hao Yuan[2], and Mikhail J. Atallah[3]

[1] Computer Science and Software Engineering, Miami University
frikkekb@muohio.edu
[2] Department of Computer Science, City University of Hong Kong
haoyuan@cityu.edu.hk
[3] Department of Computer Science, Purdue University
mja@cs.purdue.edu

Abstract. In the third-party model for the distribution of data, the data owner provides a third party (referred to as the dealer) with data as well as integrity verification information for that data, in the form of digital signatures that the dealer can use to convince a user of the data's integrity (the dealer is not trusted with the owner's signature keys, which is why it receives pre-signed items). The user's interactions are with the dealer, who is in charge of enforcing access control and confidentiality for the data (i.e., no user should learn more than the outcome of their authorized query). This kind of outsourcing is becoming increasingly important because of its advantageous economics – a dealer who acts as a repository for many owners can achieve economies of scale that are not feasible for the individual owners, and the model allows the data owners to focus on what they do best (the creation and/or acquisition of high-quality data). A problem that arises in the context of outsourced databases (particularly for XML data) is the following: There is a total order Π on n items stored with the dealer, and a user query consists of a pair of items whose relative ordering should be revealed along with a proof that the result is correct. The proof is generated using the dealer's local data (i.e., without bothering the data owner). The main difficulty is achieving efficient storage and query-processing while achieving the desiderata (that the user should learn nothing other than the answer to their query, and that a misbehaving dealer should not be able to convince a user of a wrong ordering). This paper gives a solution that is provably secure under a new assumption and can efficiently generate a very short proof. Furthermore, this scheme is generalized to partial orders that can be decomposed into d total orders. In this case, a user either learns the ordering of the queried items, or learns that they are incomparable.

Keywords: Authenticated Outsourcing, Cryptographic Protocols.

* Portions of this work were supported by National Science Foundation Grants CNS-0915436, CNS-0913875, CNS- 0915843, Science and Technology Center CCF-0939370; by an NPRP grant from the Qatar National Research Fund; by Grant FA9550-09-1-0223 from the Air Force Office of Scientific Research; by sponsors of the Center for Education and Research in Information Assurance and Security; and by a grant from City University of Hong Kong (Project No. 7200218). The statements made herein are solely the responsibility of the authors.

J. Lopez and G. Tsudik (Eds.): ACNS 2011, LNCS 6715, pp. 514–531, 2011.
© Springer-Verlag Berlin Heidelberg 2011

1 Introduction

Critical data is nowadays accessed through the Internet, often through servers that do not belong to the data owner. The protection of such re-distributed information was explicitly mentioned by Tygar in his early summary of critical open problems in electronic commerce [42]. When important decisions are made based on such data (e.g., business, health-care, scientific, etc), it becomes especially important to provide a user the means of verifying the integrity of the data to which they are entitled; such access rights for data occur because of a user's role in their organization, their government security clearance level, or perhaps because of payment for access. A compelling case was made in the third-party distribution literature, reviewed below in Section 5, for the use of schemes that reduce the trust level required from a third-party dealer. But some thorny issues arise when data distribution is done through a third party dealer who is "moderately trusted" in the sense that, although the data owner relies on the dealer for distribution, the owner does not wish to provide the dealer with the authority to sign on its behalf; this is typically not out of fear for the signature keys (special keys can be created for that purpose), but rather for purposes of data quality control and fear of liability – a compromised dealer should not have the capability of convincing users that corrupted data is pristine. This caution is often not motivated by mistrust of the dealer as an organization, rather, it is a realistic recognition of the fact that networked systems are vulnerable to break-ins, spyware, insider misbehavior, or simply accidents.

The third-party distribution model offers many economic advantages but also a challenge: How does the third-party distributor, who does not have the authority to sign, prove to a user the integrity of the answer to the user's query? There are typically too many possible queries for the owner to pre-sign and store with the dealer a signature for each – the space taken by the data and its signatures on the dealer's system should have size that is close to linear in the size of that data. The problem is conceptually easy when a query asks for a subset of the n data items, as the use of aggregate signatures enables the dealer to convince the user with a single signature (that would be the aggregate, computed by the dealer, of the pre-stored signatures for the individual items in the user's query). But queries about semi-structured data involve comparisons, e.g., they require a proof that node v is to the left of a sibling node w without revealing to the user anything else (e.g., the number of other siblings that are between v and w in the structure); the authors of [28,29] explicitly make the case for the necessity in the XML context of carrying out sibling-order comparisons with the exact properties we require. The problem of comparisons also arises with "flat" data, when there is value-added information computed by the owner and that imposes structure on the individual items. The simplest such structure is a total ordering of the n items stored at the dealer, according to the data owner's proprietary evaluation methodology for the items; there could even be k such total orderings, each of which is based on different evaluation criteria from the others. A user who is comparing the qualifications of two or more of the items stored with the dealer will want to know which is the best for the purpose contemplated. Examples of such criteria include: For publicly traded securities, their current level of attractiveness as an investment, or how they will be affected by some future event; in geopolitical consulting, estimates how one specific hypothetical future would impact various corporate or political entities (or, more boldly, estimates of the

likelihood of each of n possible futures to materialize); for individuals or corporations offering services, their levels of proficiency, or future promise, or likelihood of misbehavior; for technologies or products, their effectiveness/reliability/cost for particular tasks. If a user is only authorized to compare two items X and Y, providing the user with the actual score of each would allow the user to compare items other than X and Y (violating disclosure rules, and possibly depriving the data owner of future revenue).

Our contribution: This paper presents an efficient and provably secure scheme for the problem of authenticated secure comparisons that was informally sketched above, and that will be more formally defined in the next section. The scheme requires an amount of storage, at the dealer, that is $O(n \log n)$ where n is the number of items, and the setup and protocols are efficient in that they take a constant amount of communication (hence rounds). We prove that our scheme does not reveal to a user anything other than the outcome of a comparison, that a user can verify the correctness of the answer, and that a rogue dealer is unable to prove a false answer to a query. The security of our scheme is proven in the standard model and is based on a new assumption over bilinear maps, which we call a Computational Linear Splitting Assumption (CLS): That it is hard for an adversary who has g^a, g^b, g^c, g^d to produce g^m and g^n such that $mn = ab + cd$ (note that a decisional version of this assumption is not true). To give evidence of the security of this assumptions, we prove CLS is secure in the generic group model.

2 Preliminaries

2.1 Notation

The security of our approach is in the computational model, and thus we assume the adversaries are probabilistic polynomial time (PPT). A function $\epsilon(n)$ is negligible if for all polynomials p, $\exists N$ such that $\forall n > N$, $\epsilon(n) < \frac{1}{p(n)}$. Given a set S, the notation $s \stackrel{U}{\leftarrow} S$ corresponds to choosing a value uniformly from S.

2.2 Problem Formulation

In our model we consider three types of entities, a data owner, a data dealer, and users. The data owner has a set of identities which we denote as $\{1, \ldots, n\}$[1] and a permutation $\Pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ that defines a total ordering of the identities (based on, e.g., some proprietary information of the data owner). The data owner would like to support queries of the form "Which is greater according to Π, identity i or identity j?". What makes this difficult is that the data owner wants to provide the answers in an outsourced, verified, and zero-knowledge manner. More specifically, the data owner does not want to provide a service that answers queries, and would like instead to outsource the answering of such queries to the data dealer. However, the users of the system do not trust that the data dealer will honestly answer queries, and thus the data dealer must provide a proof that the results are correct ("as good if the user got them directly from

[1] If the identities are not in this domain, then the data owner simply provides a mapping between the real identities and these values.

the owner"). Finally, the queries should be answered in a confidential manner such that the users do not receive additional information other than the answers of their queries as well as what can be inferred from such queries.

Formally, authenticated outsourced secure comparisons consist of three algorithms:

1. $\mathrm{SETUP}(\Pi, 1^\kappa)$ takes as input the ordering Π of the identities and a security parameter 1^κ. The output is (dk, pk) which are respectively the dealer key and the public key.
2. $\mathrm{PROVE}(i, j, pk, dk)$ takes as input identities i and j (such that $\Pi(i) < \Pi(j)$), the public key, pk, and the dealer key, dk. The output is a certificate $cert$.
3. $\mathrm{VRFY}(i, j, pk, cert)$ takes as input identities i and j, the public key, pk, and a certificate $cert$. The output of this algorithm is a binary value b. If $b = 1$ we say the the certificate is accepted, otherwise we say that the certificate is rejected.

2.3 Security Definitions

The security goals of this paper are threefold: i) correctness, ii) protection against dishonest dealer, and iii) zero-knowledge queries.

Correctness. The correctness requirement simply states that if the dealer is honest then the verification algorithm accepts the certificates. More formally, given permutation Π and security parameter κ, then $\forall i, j \in [1, n]$ such that $\Pi(i) < \Pi(j)$:

$$\Pr\big[\mathrm{VRFY}(i, j, pk, cert) = 1 : (dk, pk) \leftarrow \mathrm{SETUP}(\Pi, 1^\kappa), cert \leftarrow \mathrm{PROVE}(i, j, dk, pk)\big] = 1.$$

Protection against Dishonest Dealer. This requirement states that a dishonest dealer cannot convince an honest user that $\Pi(i) < \Pi(j)$ when in fact $\Pi(j) < \Pi(i)$. For a specific protocol $\Lambda = (\mathrm{SETUP}, \mathrm{PROVE}, \mathrm{VRFY})$, this notion is captured in the experiment in Figure 1. Protocol Λ is secure against forgery by a dishonest dealer if, for all probabilistic polynomial time adversaries A, $\Pr[\mathbf{Exp}_{\Lambda,A}^{\text{dealer-forge}}(1^\kappa, \Pi)] = 1]$ is negligible in κ.

Zero Knowledge Queries. Let π be an oracle that provides indirect access to the ordering Π; moreover, $\pi(i, j)$ returns 1 if $\Pi(i) < \Pi(j)$ and returns 0 otherwise. Suppose algorithm SIM^π has oracle access to π. More specifically, whenever SIM needs to know the relationship about i and j it queries the oracle for $\pi(i, j)$. After several queries to π, let $\hat{G}$ be the relationship graph defined by these queries. Clearly, if there is no path between i and j (or vica versa) in $\hat{G}$, then either ordering is possible. Furthermore, SIM does not know this relationship. Informally, a scheme has zero-knowledge queries if for every PPT adversary $\mathcal{A}$, there is a PPT simulator SIM^π that can simulate the $\mathcal{A}$'s view in the real protocol.

More formally, in the real protocol $\mathcal{A}$ is given the public key pk, and then can adaptively ask for the comparison and certificate for each of the pairs $(i_1, j_1), \ldots, (i_m, j_m)$. Thus the view of adversary $\mathcal{A}$ is $(pk, res_1, cert_1, \ldots, res_m, cert_m)$ where res_i is the

Experiment $\mathbf{Exp}^{\text{dealer-forge}}_{\Lambda,A}(1^\kappa, \Pi)$
$(dk, pk) \leftarrow \text{SETUP}(1^\kappa, \Pi)$
$(i, j, cert) \leftarrow A(1^\kappa, \Pi, dk, pk)$
if $\Pi(i) > \Pi(j)$ and $\text{VRFY}(i, j, pk, cert) = 1$ then return 1
else return 0

Fig. 1. A forgery experiment involving a dishonest dealer

Experiment $\mathbf{Exp}^{\text{CLS}}_A(1^\kappa)$
$[q, \mathbb{G}, \mathbb{G}_T, g, e] \leftarrow \text{BGen}(1^\kappa)$
$a, b, c, d \overset{U}{\leftarrow} \mathbb{Z}_q$
$(\alpha, \beta) \leftarrow A(q, \mathbb{G}, \mathbb{G}_T, g, e, g^a, g^b, g^c, g^d)$
if $e(\alpha, \beta) = e(g^a, g^b) * e(g^c, g^d)$ then return 1
else return 0

Fig. 2. $\mathbf{Exp}^{\text{CLS}}_A(1^\kappa)$

result of the ith comparison, and we denote this view as $\text{VIEW}^{\Lambda}_{\mathcal{A}}(1^\kappa, \Pi)$. In the simulated protocol $\mathcal{A}$ interacts with a simulator SIM^{π}, which must produce a public key and then adaptively provide comparisons and certificates for pairs $(i_1, j_1), \ldots, (i_m, j_m)$. Let $\text{SIM}^{\pi}_{\mathcal{A}}(1^\kappa)$ be the values output by a simulator when asked queries by adversary $\mathcal{A}$.

We say that Λ provides perfect zero knowledge queries if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator SIM such that $\text{VIEW}^{\Lambda}_{\mathcal{A}}(1^\kappa, \Pi)$ and $\text{SIM}^{\pi}_{\mathcal{A}}(1^\kappa)$ are identically distributed.

2.4 Bilinear Maps and Assumptions

We utilize bilinear maps in our protocol. Specifically, we utilize a method $\text{BGen}(1^\kappa)$ that produces $\mathbf{G} = [q, \mathbb{G}, \mathbb{G}_T, g, e]$ where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order q, q is a prime with κ bits, g is a generator of the group $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a non-degenerate mapping where the bilinear property holds. That is, $e(g^r, g^s) = e(g, g)^{rs}$.

In this paper we propose a new assumption (which is proven secure in the generic group model in the Appendix), called the Computational Linear Splitting (CLS) assumption. At a high level, CLS states that if an adversary is given $\mathbf{G}, g, g^a, g^b, g^c, g^d$, then it is hard for the adversary to produce g^m and g^n such that $mn = ab + cd$. Clearly, a decisional version of this assumption is not true. That is, given g^m and g^n it is easy to verify whether $mn = ab + cd$. This is done by simply checking whether $e(g^m, g^n) = e(g^a, g^b) * e(g^c, g^d)$. Our protocols take advantage of this gap between the computational and decisional versions of these problems, by requiring the prover to create such a pair of values, which can then be verified by the user. More formally, the CLS experiments is defined in Figure 2. Our assumption is that for all probabilistic polynomial time adversaries A, $\Pr[\mathbf{Exp}^{\text{CLS}}_A(1^\kappa) = 1]$ is negligible in κ. It is useful to compare this assumption to the well-known Computational Diffie-Hellman (CDH) assumption, which states that it is difficult to computer g^{ab} when given g^a and g^b. Obviously, if the CDH problem is easy, then so is the CLS problem. However, the converse does not appear to be true since the adversary has the flexibility of choosing any m and n values.

3 Scheme

Initially we introduce two straightforward solutions to the problem, based on signatures. We then proceed to the main construction of the paper.

3.1 Two Straightforward Schemes

Suppose that $(\mathrm{KeyGen}, \mathrm{Sign}, \mathrm{Vrfy})$ is a standard signature scheme. A simple solution to this problem is to set the public key to the verification key, and $\forall i, j \in [1, n]$ the owner signs the message $i||j$ (resp $j||i$) if $\Pi(i) < \Pi(j)$ (resp $\Pi(j) < \Pi(i)$). The dealer is then simply given all of these signatures. When the user asks for a proof about a pair of values the dealer simply provides the signature from the owner. Now, forging a proof is as hard as forging a signature, and confidentiality follows because all that is revealed is the result. The downside with this approach is that the dealer must store $O(n^2)$ values which limits the scalability of this approach.

Another approach takes the following form: The owner creates commitments for the rank of each items and signs messages of the form $(i||\mathrm{Commit}(\Pi(i)))$ for all $i \in [1, n]$ and sends these signatures to the dealer. When the user asks for the comparison of i and j, the dealer reveals the corresponding commitments and their signatures along with a zero knowledge proof that the value held by one commitment is smaller than the other value. We delay an exact comparison of this approach to our approach until section 5.1.

3.2 A More Interesting Scheme

The two key ideas in our scheme are (i) the use of dummy values each with a number of pre-signed comparison statements about how each dummy value compares relative to some of the identities; (ii) the randomization of the signed statements involving the dummies (so that the user cannot recognize if the same dummy is involved in two comparisons). The dummy values are set up in a sorted binary tree. The root of the tree will be used to prove that any identity in the left half of the tree is smaller than any identity in the right half of the tree. The rest of the internal nodes of the tree are used to prove relationships between nodes in the same subtree. A crucial property that we exploit is that if $\Pi(i) < \Pi(j)$, then there will be exactly one node in the binary tree such that i is in the left subtree of the node and j is in the right subtree; hiding which node is being so used, is necessary to prevent extra information from being revealed.

The algorithms are as follows:

- SETUP Run $\mathrm{BGen}(1^\kappa)$ to obtain $\mathbf{G} = [q, \mathbb{G}, \mathbb{G}_T, g, e]$. Choose $2n+2$ values from $\mathbb{Z}_q$ and denote them by $s, s', s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^-$ and $n-1$ values $t_1, \ldots, t_{n-1}$ from $\mathbb{Z}_q^\star$. Let $\hat{T}$ be a complete binary tree built on top of the sorted (according to Π) version of $[1, n]$, that is, the leftmost (resp., rightmost) leaf of $\hat{T}$ is the i such that $\Pi(i) = 1$ (resp., $\Pi(i) = n$). Denote the non-leaf nodes of $\hat{T}$ as $v_1, \ldots, v_{n-1}$. For each non-leaf node v_j of $\hat{T}$ the following is done: For every i in the left subtree[2] of v_j in $\hat{T}$, the owner creates a value $R_{i,j} = g^{\frac{ss_i^+}{t_j}}$. For every i in the right subtree

[2] Where the "left subtree" of a node is the subtree of the node's left child.

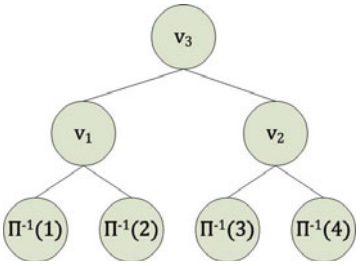

Fig. 3. Example Tree

of v_j in $\hat{T}$, the owner creates a value $S_{i,j} = g^{\frac{s's_i^-}{t_j}}$. At a high level $R_{i,j}$ (resp. $S_{i,j}$) will be a "certificate" that i is in the left (resp., right) subtree of v_j. Denote all of these R values (resp., S values) for all vertices as the set $\hat{R}$ (resp., $\hat{S}$).
The respective keys produced are as follows:

- **Public Key pk:** $\mathbf{G}, g^s, g^{s'}, g^{s_1^+}, \ldots, g^{s_n^+}, g^{s_1^-}, \ldots, g^{s_n^-}$
- **Dealer Key dk:** $\hat{T}, \Pi, \hat{R}, \hat{S}, g^{t_1}, \ldots, g^{t_{n-1}}$

– PROVE When given i, j, pk, and dk such that $\Pi(i) < \Pi(j)$, the dealer finds the unique vertex $v_k \in \hat{T}$ such that i is in the left subtree of v_k and j is in the right subtree of v_k. Choose a random α from $\mathbb{Z}_q^\star$ and set $cert := ({R_{i,k}}^\alpha * {S_{j,k}}^\alpha, (g^{t_k})^{\alpha^{-1}}) = (g^{\frac{\alpha(ss_i^+ + s's_j^-)}{t_k}}, g^{\frac{t_k}{\alpha}})$.
– VRFY When given i, j, pk, and $cert = (\beta, \gamma)$, then check if $e(\beta, \gamma) = e(g^s, g^{s_i^+}) * e(g^{s'}, g^{s_j^-})$. Output 1 if this check is true and output 0 otherwise.

Example: Suppose that $n = 4$, the tree created by the above algorithm is shown in Figure 3. For this example, the R and S values that would be part of the dealer key are as follows: $R_{\Pi^{-1}(1),1}$, $R_{\Pi^{-1}(3),2}$, $R_{\Pi^{-1}(1),3}$, $R_{\Pi^{-1}(2),3}$, $S_{\Pi^{-1}(2),1}$, $S_{\Pi^{-1}(4),2}$, $S_{\Pi^{-1}(3),3}$, and $S_{\Pi^{-1}(4),3}$. Let $x = \Pi^{-1}(2)$ and $y = \Pi^{-1}(3)$. To prove that $\Pi(x) < \Pi(y)$, the PROVE protocol would use node v_3 which separates the two nodes in the tree. That is, the certificate would be: $({R_{\Pi^{-1}(2),3}}^\alpha * {S_{\Pi^{-1}(3),3}}^\alpha, (g^{t_3})^{\alpha^{-1}})$.

3.3 Complexity Analysis

The parameters of interest are: i) setup complexity: $O(n \log n)$, ii) size of public key: $O(n)$, iii) size of the dealer key: $O(n \log n)$, iv) complexity of proof: $O(1)$, v) proof size: $O(1)$, and vi) verification cost: $O(1)$. The setup complexity, public key size, dealer key size, proof size, and verification cost all are straightforward from the protocol. It would appear that proof creation requires $O(\log n)$ complexity because it must find the dummy node that separates the leaf nodes corresponding to i and j. However, this node is simply the nearest common ancestor (NCA) of these leaf nodes. It is possible to pre-process the tree in linear time to allow finding the NCA in $O(1)$ time using the data structure of Harel and Tarjan [25]. Note that this pre-processing would be done during the setup.

Reducing the Size of the Public Key to $O(1)$. With a slight increase in the proof size (and in the verification cost), it is possible to reduce the public key size to $O(1)$. The key to this modification is that the querier only needs to know the values $\mathbf{G}, g^s, g^{s'}, g^{s_i^+}$, and $g^{s_j^-}$ when verifying the proof for $\Pi(i) < \Pi(j)$. The linear part of the key is the $+$ and $-$ values. In the modified scheme the public key consists of $\mathbf{G}, g^s, g^{s'}, pk$ where pk is a public key associated with a signature scheme. The owner signs each piece (along with meta-information about the component) of the original public key with its signature key and gives it to the dealer. When the dealer creates the proof, it simply sends the needed pieces (along with their signatures) to the verifier. More formally, the dealer appends the following values to the proof: $(i, +, g^{s_i^+}), Sign_{pk}(i|| + ||g^{s_i^+})$ and $(j, -, g^{s_j^-}), Sign_{pk}(j|| - ||g^{s_j^-})$.

3.4 Proof of Security

The correctness follows from: let $(dk, pk) \leftarrow \text{SETUP}(\Pi, 1^\kappa)$, let i and j be two identities such that $\Pi(i) < \Pi(j)$, and let $(\beta, \gamma) \leftarrow \text{PROVE}(i, j, dk, pk)$. Now, $e(\beta, \gamma) = e(g^{\frac{\alpha(ss_i^+ + s's_j^-)}{t_k}}, g^{\frac{t_k}{\alpha}}) = e(g,g)^{ss_i^+ + s's_j^-} = e(g^s, g^{s_i^+}) * e(g^{s'}, g^{s_j^-})$.

Security against Dishonest Dealer. Suppose that the scheme, Λ, is not secure against a dishonest dealer; that is there exists a polynomial-time adversary $\mathcal{A}$ such that $\Pr[\mathbf{Exp}^{\text{dealer-forge}}_{\Lambda,\mathcal{A}}(1^\kappa, \Pi)] = 1] > \frac{1}{p(\kappa)}$ for some polynomial $p(\kappa)$. We will show how to construct an adversary $\mathcal{B}$ with black box access to $\mathcal{A}$ that solves the CLS with non-negligible probability.

Given an instance of CLS, we are given $\mathbf{G} = [q, \mathbb{G}, \mathbb{G}_T, g, e]$ and g^a, g^b, g^c, and g^d. Algorithm $\mathcal{A}$ will produce a forgery for some pair of values, but as the choice of which pair of values is unknown, algorithm $\mathcal{B}$ will randomly pick two values $\hat{i}, \hat{j}$ such that $\Pi(\hat{j}) < \Pi(\hat{i})$. We will ensure that the public key and dealer key given to $\mathcal{A}$ will be distributed identically to the real protocol, and since there are only $O(n^2)$ pairs of values, $\mathcal{B}$ will guess which pair that $\mathcal{A}$ will produce a forgery with non-negligible probability.

$\mathcal{B}$ chooses values $r_1^+, \ldots, r_n^+, r_1^-, \ldots, r_n^-, r_1, \ldots, r_{n-1}$ uniformly from $\mathbb{Z}_q^\star$. We implicitly set $s = a$, $s' = c$, $s_{\hat{i}}^+ = b$ and $s_{\hat{j}}^- = d$. Note that a forgery proving that $\Pi(\hat{i}) < \Pi(\hat{j})$, would require creating two values g^e and g^f such that $e(g^e, g^f) = e(g,g)^{ab+cd}$, which solves the CLS problem.

Moreover, we implicitly set

$$s_k^+ = \begin{cases} cr_k^+ & : \ \Pi(k) < \Pi(\hat{i}) \\ b & : \ \Pi(k) = \Pi(\hat{i}) \\ r_k^+ & : \ \Pi(k) > \Pi(\hat{i}) \end{cases} \quad s_k^- = \begin{cases} r_k^- & : \ \Pi(k) < \Pi(\hat{j}) \\ d & : \ \Pi(k) = \Pi(\hat{j}) \\ ar_k^- & : \ \Pi(k) > \Pi(\hat{j}) \end{cases}$$

Given a tree vertex v_k, denote as ℓ_k the rightmost leaf node in the left subtree of v_k. If $\Pi(\hat{j}) \leq \Pi(\ell_k)$ we set $t_k = ar_k$ and otherwise we set $t_k = cr_k$. It is important to notice that we cannot generate many of the above values, but we can generate all

$\mathcal{B}(1^\kappa, \mathbf{G}, g^a, g^b, g^c, g^d)$
$\hat{i}, \hat{j} \overset{U}{\leftarrow} [1, n]$
$(pk, dk) \leftarrow \mathrm{PGen}(1^\kappa, \Pi, \hat{i}, \hat{j})$
$(i, j, cert = (\alpha, \beta)) \leftarrow \mathcal{A}(1^\kappa, \Pi, dk, pk)$
If $i = \hat{i}$ and $j = \hat{j}$ output α, β
else output FAIL

Fig. 4. Algorithm $\mathcal{B}$

values $g^{s_k^+}$, $g^{s_k^-}$, and g^{t_k}. Furthermore, each s_k^+, s_k^-, and t_k are distributed identically to the values chosen in the protocol. All that is left to show is how to compute $R_{k,m}$ and $S_{k,m}$. Let k be an identity in the left subtree of v_m, then we need to be able to compute $R_{k,m} = g^{\frac{ss_k^+}{t_m}} = g^{\frac{as_k^+}{t_m}}$, there are two cases to consider: i) $\Pi(\hat{j}) \leq \Pi(\ell_m)$: In this case $t_m = ar_m$, and thus $R_{k,m} = g^{\frac{as_k^+}{ar_m}} = g^{\frac{s_k^+}{r_m}}$, and ii) $\Pi(\hat{j}) > \Pi(\ell_m)$: Since $\hat{j}$ is after ℓ_m, then $t_m = cr_m$ and $\Pi(k) \leq \Pi(\ell_m) < \Pi(\hat{j}) < \Pi(\hat{i})$, thus $s_k^+ = cr_k^+$, and hence, $R_{k,m} = g^{\frac{acr_k^+}{cr_m}} = g^{\frac{ar_k^+}{r_m}}$.

Let k be an identity in the right subtree of v_m, then we need to be able to compute $S_{k,m} = g^{\frac{s's_k^-}{t_m}} = g^{\frac{cs_k^-}{t_m}}$, there are two cases to consider: i) $\Pi(\hat{j}) \leq \Pi(\ell_m)$: In this case $t_m = ar_m$. Also, since k is in the right subtree of v_m, we know that $\Pi(\hat{j}) < \Pi(k)$, thus $s_k^- = ar_k^-$, and hence, $S_{k,m} = g^{\frac{car_k^-}{ar_m}} = g^{\frac{cr_k^-}{r_m}}$, and ii) $\Pi(\hat{j}) > \Pi(\ell_m)$: In this case $t_m = cr_m$, and thus $S_{k,m} = g^{\frac{cs_k^-}{cr_m}} = g^{\frac{s_k^-}{r_m}}$.

Let $\mathrm{PGen}(1^\kappa, \Pi, \hat{i}, \hat{j})$ be the procedure described above for computing pk and dk. Algorithm $\mathcal{B}$ works as described in Figure 4.

If $\mathcal{B}$ does not output FAIL and $\mathcal{A}$ succeeds in producing a forgery, then $\mathcal{B}$ will solve the CLS. Now,

$$\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathrm{CLS}}(1^\kappa) = 1] = \Pr[\mathbf{Exp}_{\Lambda,\mathcal{A}}^{\text{dealer-forge}}(1^\kappa, \Pi) = 1 \wedge \mathcal{B}(1^\kappa, \mathbf{G}, g^a, g^b, g^c, g^d) \neq FAIL]$$

Regardless of $\mathcal{B}$'s choice for $\hat{i}$ and $\hat{j}$ the public key and dealer key produced by $\mathcal{B}$ is distributed identically. Thus these two events are independent, and the former's probability is $\geq \frac{1}{p(\kappa)}$ and the latter's probability is $\geq \frac{1}{n^2}$. Thus $\Pr[\mathbf{Exp}^{\mathrm{CLS}_{\mathcal{B}}}(1^\kappa) = 1] \geq \frac{1}{n^2 p(\kappa)}$ which is not negligible. □

Zero Knowledge Queries. The proof of zero knowledge queries rests on the certificates produced by $\mathrm{PROVE}(i, j, pk, dk)$ being distributed identically to tuples of the form $\{g^{r^{-1}(ss_i^+ + s's_j^-)}, g^r) : r \leftarrow \mathbb{Z}_q^\star\}$. This follows because the certificate is of the form $({R_{i,k}}^\alpha * {S_{j,k}}^\alpha, (g^{t_k})^{\alpha^{-1}}) = (g^{\frac{\alpha(ss_i^+ + s's_j^-)}{t_k}}, g^{\frac{t_k}{\alpha}})$ where $\alpha \overset{U}{\leftarrow} \mathbb{Z}_q^\star$. Let $\rho = \frac{t_k}{\alpha}$, in this case the certificate is simply $(g^{\rho^{-1}(ss_i^+ + s's_j^-)}, g^\rho)$. Now ρ and r are distributed identically, because α is a randomly chosen value uniformly from $\mathbb{Z}_q^\star$. Thus the claim follows.

Given the above observation, the simulator will operate as follows: it will do the same random setup to produce its own pk, dk. The public key will be chosen identically to that in the real protocol (since it is generated in the same manner). A difference is that the simulator knows the values of s, s', s_1^+, s_1^-, etc. And so, the simulator can prove any relationship between any pairs of values. When the client asks for a specific proof, the simulator gets the correct response using its oracle to π and then produces a proof that is distributed identically to the real proof. The specific simulator, $\mathrm{SIM}_{\mathcal{A}}^{\pi}(1^{\kappa})$, is described formally below:

Simulator Setup: The simulator does the following steps:

1. $\mathbf{G} \leftarrow \mathrm{BGen}(1^{\kappa})$
2. $s, s', s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^- \overset{U}{\leftarrow} \mathbb{Z}_q$
3. $pk := \mathbf{G}, g^s, g^{s'}, g^{s_1^+}, \ldots, g^{s_n^+}, g^{s_1^-}, \ldots, g^{s_n^-}$
4. The simulator gives $\mathcal{A}$ the public key pk.

Query responses: When $\mathcal{A}$ queries for the relationship between i and j, the simulator uses its oracle access to π to determine the relationship between the two values. Without loss of generality suppose that $\Pi(i) < \Pi(j)$. The simulator then does the following: Choose $r \overset{U}{\leftarrow} \mathbb{Z}_q^{\star}$, and return $(g^{r^{-1}(ss_i^+ + s's_j^-)}, g^r)$.

Now, $\mathrm{VIEW}_{\mathcal{A}}^{\Lambda}(1^{\kappa}, B_1, \ldots, B_n, \Pi)$ consists of pk and several certificates. Now pk is generated by the same process in both the simulator and the real protocol, and thus these are distributed identically. Furthermore, the certificates also have the same distribution. Thus these views are distributed identically. □

4 Extensions

4.1 Min/Max Queries

Consider the following authenticated range minimum query (RMQ) problem: a user wants to query for the position of the minimum element among $\Pi(i), \Pi(i+1), \Pi(i+2), \cdots, \Pi(j)$, where $i < j$ are two query parameters. We use $\mathrm{RMQ}(i, j)$ to denote the position k such that $\Pi(k) = \min_{i \le p \le j} \Pi(p)$.

To return authenticated $k = \mathrm{RMQ}(i, j)$ without leaking Π, a naive method is: For each $p \in [i, k-1] \cup [k+1, j]$, the dealer sends to the user a proof that $\Pi(k) < \Pi(p)$. This naive approach has a complexity of $O(n)$. More efficient solution can be obtained using the technique of Cartesian trees [43]. In the Cartesian tree for Π, each tree node v_i has two keys: i and $\Pi(i)$. The tree is organized in a unique way that: it is a binary search tree under the first keys, and a minimum heap under the second keys. In Gabow, Bentley and Tarjan [18], it was shown that the Cartesian tree for Π can be constructed in $O(n)$ time, and $\mathrm{RMQ}(i, j) = \mathrm{LCA}(i, j)$, where $\mathrm{LCA}(i, j)$ is the least (or lowest) common ancestor of nodes v_i and v_j in the Cartesian tree. Therefore, it is sufficient to prove that v_k (where $k = \mathrm{RMQ}(i, j)$) is the lowest common ancestor of v_i and v_j, with the requirement that the structure of the Cartesian tree is not leaked.

To prove that node v_k is the lowest common ancestor of v_i and v_j, it is equivalent to prove that:

1. v_k is an ancestor of v_i;
2. v_k is an ancestor of v_j;
3. v_k's left child (if it exists) is not an ancestor of v_j;
4. v_k's right child (if it exists) is not an ancestor of v_i.

Property (1) and (2) can be easily tested in the following way [39,26]: A node u is a proper ancestor of a node v if and only if the preorder number [27] of u is smaller than the preorder number [27] of v, and the postorder number of u is bigger than the postorder number of v. Therefore, the owner can define two new mappings $\Pi_1(i) =$ preorder number of v_i, and $\Pi_2(i) =$ postorder number of v_i, and then let the dealer to run the authenticated secure comparison protocol to prove (1) and (2) based on Π_1 and Π_2. Property (3) can be tested by showing that the postorder number of the left child of v_k is smaller than the postorder number of the left child of v_j. Note that we can add one or two dummy children as new leaves to each tree node to make sure each tree node has two children. Similarly, property (4) can be tested by comparing the preorder numbers of the right children of v_k and v_i. To generate efficient proofs for property (3) and (4), let T' be the new Cartesian tree with added dummy leaves, then the owner can define two new mappings Π_3 and Π_4, where $\Pi_3(i)$ is the preorder number of the left child of v_i in T', and $\Pi_4(i)$ is the postorder number of the right child of v_i in T'. The dealer then runs the authenticated secure comparison protocol to prove property (3) and (4) based on Π_3 and Π_4. The complexity for proving the LCA and RMQ is then reduced to $O(1)$.

4.2 Extension to Partial Orders

Given a partial order, it is possible to decompose the partial order into the intersection of a group of t total orders, and the dimension of a partial order is the minimum such t. In this section we describe a technique for extending the system to d-dimensional partial orders where the decomposition into d total orders is known. Unfortunately, computing the dimension of a partial order is NP-complete [44]. However, there are some cases where computing the dimension (and the decomposition) is efficient. For example, trees have dimension at most 2, and the decomposition is straightforward. Also, partial orders whose transitive reduction is a planar graph have dimension at most 3, and the decomposition can be computed in linear time [40]. Furthermore, if the transitive reduction is 4-colorable then the dimension is at most 4 [40]. Thus the following section extends the previous scheme to any partial order where the decomposition into total orders is known, which is significantly more general than the protocol in section 3.2. It is important to note that the following scheme does not need a minimal decomposition, however a non-minimal decomposition leads to a less efficient scheme.

Specifically, we represent the partial order over $[1, n]$ as d total order $\Pi_1, \ldots, \Pi_d$. Moreover, if $\bigwedge_{i=1}^{d} \Pi_i(k) < \Pi_i(j)$, then we say that k is less than j. However, if $\exists i_1, i_2 \in [1, d] : \Pi_{i_1}(k) < \Pi_{i_1}(j)$ and $\Pi_{i_2}(j) < \Pi_{i_2}(k)$, then we say that j and k are incomparable. Proving statements of the form "k is less than j" is straightforward, we utilize d versions of the scheme outlined in section 3.2, one for each of the d dimensions. If "k is less than j', then the dealer will be able to prove this statement for all dimensions. The interesting part of this protocol is creating a proof that two elements

are incomparable. If j and k are incomparable, then there will be a pair, Π_{d_1} and Π_{d_2} of partial orders where $\Pi_{d_1}(k) < \Pi_{d_1}(j)$ and $\Pi_{d_2}(j) < \Pi_{d_2}(k)$, and thus it would appear that all that needs to be done is to use the schemes for dimensions d_1 and d_2 to prove both of these statements. Unfortunately, this reveals additional information, namely it would reveal the specific dimensions to the querier. To hide this information we will apply the scheme to all d-dimensions a second time, and furthermore use the same values $s, s', s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^-$ for each dimension. With this approach the dealer will be able to prove that "k is less than j" and that "j is less than k" for incomparable values. However, for comparable values the dealer would not be able to generate such proofs.

The scheme requires $O(d)$ versions of the original protocol. Thus the size of the public key is $O(dn)$ and the size of the dealer key is $O(dn \log n)$. The size of a proof that $\Pi(i) < \Pi(j)$ is $O(d)$ and the size of a proof of incompatibility is $O(1)$. Note that a zero-knowledge incompatibility proof based on range proof protocols for commitments would result in a proof size of $O(d)$. Therefore, our scheme has a significant advantage for proving incompatibility efficiently.

An authenticated outsourced private secure comparisons for partial orders consists of five algorithms ($POSETUP, POPROVELT, POPROVEIN, POVRFYLT$, $POVRFYIN$), which correspond to setup, proving less than, proving incomparability, verifying less than, and verifying incomparability.

Before describing the scheme formally, we define two variations of the Setup protocol in section 3.2. These variations allow us to specify some of the parameters used by the setup algorithm.

1. $\mathrm{SETUP}(\mathbf{G}, \Pi, 1^\kappa)$: This does all of the steps in setup, but uses the group $\mathbf{G}$ instead of creating its own group.
2. $\mathrm{SETUP}(\mathbf{G}, \Pi, 1^\kappa, s, s', s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^-)$: This does all of the steps in SETUP, but uses the group $\mathbf{G}$ instead of creating its own group, and uses the values $s, s', s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^-$ instead of generating its own parameters.

The actual protocols are as follows:

- POSETUP Run $\mathrm{BGen}(1^\kappa)$ to obtain $\mathbf{G} = [q, \mathbb{G}, \mathbb{G}_T, g, e]$. For each $j = 1$ to d run $\mathrm{SETUP}(\mathbf{G}, \Pi_j, 1^\kappa)$ and denote the keys as (pk_j, dk_j).

  Furthermore, choose $2n+2$ values from $\mathbb{Z}_q$ and denote them by $s, s', s_1^+, \ldots, s_n^+$, $s_1^-, \ldots, s_n^-$ For each $j = 1$ to d run $\mathrm{SETUP}(\mathbf{G}, \Pi_j, 1^\kappa, s, s', s_1^+, \ldots, s_n^+, s_1^-, \ldots, s_n^-)$ and denote the keys as $(\hat{pk}_j, \hat{dk}_j)$. Note that $\hat{pk}_j$ is the same for all dimensions $j \in [1, d]$, and thus it is denoted simply as $\hat{pk}$. The keys defined by setup are as follows:
  - **Public Key pk:** $pk_1, \ldots, pk_d, \hat{pk}$
  - **Dealer Key dk:** $dk_1, \ldots, dk_d, \hat{dk}_1, \ldots, \hat{dk}_d$
- POPROVELT When given i, j, pk, and dk such that i is less than j. For all $\ell \in [1, d]$, $\Pi_\ell(i) < \Pi_\ell(j)$. Thus for each dimensions $\ell \in [1, d]$, the dealer creates $cert_\ell = \mathrm{PROVE}(i, j, pk_\ell, dk_\ell)$. The certificate is $cert_1, \ldots, cert_d$.
- POVRFYLT When given i, j, pk, and $cert = cert_1, \ldots, cert_d$, the verification algorithm checks whether $\mathrm{VRFY}(i, j, pk, cert_\ell)$ accepts for all dimensions $\ell \in [1, d]$. If so, then this outputs 1 and accepts the certificate, and otherwise this rejects the certificate and outputs 0.

- POPROVEIN When given i, j, pk, and dk such that i is incomparable with j. There exists total orders Π_{d_1} and Π_{d_2} such that $\Pi_{d_1}(i) < \Pi_{d_1}(j)$ and $\Pi_{d_2}(j) < \Pi_{d_2}(i)$. The dealer creates $cert_{i,j} = \text{PROVE}(i, j, \hat{pk}, dk_{d_1})$ and $cert_{j,i} = \text{PROVE}(j, i, \hat{pk}, dk_{d_2})$. The certificate is $(cert_{i,j}, cert_{j,i})$.
- POVRFYIN When given i, j, pk, and $cert = (cert_{i,j}, cert_{j,i})$, the verification algorithm checks whether $\text{VRFY}(i, j, \hat{pk}, cert_{i,j})$ and $\text{VRFY}(j, i, \hat{pk}, cert_{j,i})$. If both accept, then this outputs 1 and otherwise it outputs 0.

5 Related Work

Hacigümüş [24] proposed the database outsourcing framework. Devanbu et al. [15] considered the query result integrity verification problem for basic operators (e.g., Selection, Projection, Union, etc) in outsourced databases, based on the Merkle hash tree technique [33]. Mykletun et al. [34] used signature aggregation approaches (Condensed-RSA and BGLS [3]) to authenticate the SELECT query result. Narasimha and Tsudik [35] used signature aggregation and chaining to verify the query result completeness for basic operators in outsourced databases. Cheng, Pang and Tan [10] also used a similar signature chain approach to verify the completeness of multi-dimensional query results. Their signature chain approaches [35,10] to verify query result completeness leaks some data tuples outside the query result. Pang and Tan [36] proposed a repeated hashing approach to solve the data leakage problem in verifying query result completeness. Their approach can also handle range aggregate operators (like COUNT, SUM and MAX/MIN). Using a similar method, Cheng and Tan [11] considered the problem of authenticating k nearest neighbor queries. Chen et al. [9] addressed the completeness verification problem in a different access control model. Although the scheme of Pang and Tan [36] can be used to authenticate range aggregate query result, their approach leaks information other than the query results. Haber et al. [23] also provided schemes to authenticate query results, but their schemes still leak information other than the query results. Other related work on database outsourcing include authenticate dynamic outsourced databases [30,20], authenticated data structures [19,21,37], and query result freshness [30].

For the secure authenticated comparison problem, the closest work is range proof [32,16,4,31,22,5,45,8]: Given a commitment of a secret, a range proof is a zero-knowledge protocol to prove that the committed secret belongs to a specific range. Using range proof, one can solve the secure authenticated comparison problem in the following way: Let $\text{Commit}(x, r)$ represent a commitment of a secret x, where r is a random number. The commitment scheme can be either Pedersen commitment [38] or Fujisaki-Okamoto commitment [17,14], depending on the range proof protocol used.

Note that those commitment schemes are homomorphic, i.e.,

$\text{Commit}(x_1, r_1)/\text{Commit}(x_2, r_2) = \text{Commit}(x_1 - x_2, r_1 - r_2)$. The owner signs the commitments $\text{Commit}(\Pi(i), r_i)$ of $\Pi(i)$ for every i, makes them public, and then sends r_i's to the dealer. Whenever a user asks for comparing $\Pi(i)$ and $\Pi(j)$, the dealer gives a zero-knowledge proof to convince the user that the secret hidden by $\text{Commit}(\Pi(i), r_i)$ is smaller or bigger than the secret hidden by $\text{Commit}(\Pi(j), r_j)$ using a range proof. For example, to prove that $\Pi(i) < \Pi(j)$, one can prove that the secret in $\text{Commit}(\Pi(j), r_j)/\text{Commit}(\Pi(i), r_i)$ is in the range $[1, n-1]$.

The complexity for generating the range proof (in terms of modular exponentiations and final proof size) is $O(\log n)$ if the range proof protocol is based on the classic bit-by-bit approach [32,16]. Note that it is possible to make the proof size constant using the scheme of Boudot [4], Lipmaa [31], Groth [22], or Yuen et al. [45]. Note that the proof size generated by the protocol of Camenisch, Chaabouni and shelat [5] (later improved by Chaabouni, Lipmaa and shelat [8]) can also be constant for the secure authenticated comparison problem (see the following subsection). In the following section, we argue that the range proof based approach is less efficient than our scheme for the secure authenticated comparison problem.

5.1 A More Exact Comparison with Previous Work

Our protocol is non-interactive *without assuming the random oracle model* [1,7]. Most of the previous protocols [4,31,22,5] are 3-round Σ-protocols [12] if the random oracle model is not allowed. Moreover, they are *honest-verifier* zero-knowledge unless assuming random oracle model or applying some general transformation (e.g., the 4-round general zero-knowledge transformation for certain Σ-protocols by Cramer et al. [13]). The only previous range proof that is non-interactive without random oracle is due to Yuen et al. [45].

Now consider the exact proof sizes of our protocol and the previous ones. We will use $|G|$ to denote the number of bits to represent a group element in G. We follow the parameters in Camenisch, Chaabouni and shelat [5], e.g., $|G| = 256$, $|G_T| = 3072$, and $|Z_p| = 256$ for the bilinear groups G, G_T of prime order p.

In our protocol, only two group elements in G are sent to the user, so the proof size is $2|G| = 512$ bits.

Using the protocol of Camenisch, Chaabouni and shelat [5], the most efficient proof size is achieved when $u = n$ and $l = 1$ in their paper (recall that their basic protocol proves that a committed secret lies in $[0, u^l)$). The solution is sketched below.

- SETUP: The owner sends $n-1$ Boneh-Boyen signatures [2] (secure against chosen message attack) to the dealer, the signatures are for messages $1, 2, 3, \cdots, n-1$. For each $1 \leq i \leq n$, the owner commits the rank $\Pi(i)$ to C_i, signs C_i using any signature scheme, and publishes C_i and its signature.
- PROVE: To show that $\Pi(i) < \Pi(j)$, the dealer can prove in zero knowledge that it possesses a signature for the secret in C_j/C_i. This is sound because C_j/C_i is a commitment of $\Pi(j) - \Pi(i)$, which belongs to the interval $[1, n-1]$. If the dealer wants to fake a proof, then it needs to forge a signature for some integer in $[-n+1, -1]$.

The proof size by this approach is 4608 bits. Note that Camenisch and Lysyanskaya's signature scheme [6] also has an efficient signature possession proof that can be used in a way similar to the above scheme, but it is less efficient than the one by Camenisch, Chaabouni and shelat [5].

Other approaches by Boudot [4], Lipmaa [31], Groth [22] are much less efficient in the query stage, because of the needs to use 3072-bit RSA keys to match the security of our protocol and Camenisch et al.'s [5]. For example, under the random oracle model, Lipmaa's sum of four squares protocol [31] requires about $30720 + \frac{5}{2}\log_2(n-1)$ bits

to do the non-negativeness proof. Groth's sum of three squares improvement [22] requires about $23936 + 2\log_2(4n-3)$ bits. The protocol of Yuen et al. [45] requires 27648 bits, whose size is more than 50 times larger than our 512-bit proof size.

Therefore, our protocol can be viewed as a non-interactive protocol that uses much smaller query-stage proof size, while increasing server space from $O(n)$ to $O(n \log n)$.

6 Conclusion/Future Work

In this paper we give a protocol for outsourced comparisons, where the dealer has to prove that the answers are correct and all that is revealed to the querier is the outcome of the comparison queries. We give a protocol for comparisons over total orders for n items, where the proof size is $O(1)$ and the dealer is required to store $O(n \log n)$ values. The security of our approach is based on a new assumption, the Computational Linear Splitting Assumption, which may be of independent interest.

Acknowledgments

The authors thank the anonymous reviewers for their comments and useful suggestions.

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS 1993: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM, New York (1993)
2. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
3. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
4. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
5. Camenisch, J.L., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
6. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology. J. ACM 51(4), 557–594 (2004) (revisited)
8. Chaabouni, R., Lipmaa, H., Shelat, A.: Additive combinatorics and discrete logarithm based range protocols. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 336–351. Springer, Heidelberg (2010), `http://eprint.iacr.org/`
9. Chen, H., Ma, X., Hsu, W., Li, N., Wang, Q.: Access control friendly query verification for outsourced data publishing. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 177–191. Springer, Heidelberg (2008)

10. Cheng, W., Pang, H., Tan, K.: Authenticating multi-dimensional query results in data publishing. In: Proceedings of Data and Applications Security XX, 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pp. 60–73 (2006)
11. Cheng, W., Tan, K.: Authenticating knn query results in data publishing. In: Jonker, W., Petković, M. (eds.) SDM 2007. LNCS, vol. 4721, pp. 47–63. Springer, Heidelberg (2007)
12. Cramer, R.: Modular design of secure yet practical cryptographic protocols. Ph.D. thesis, CWI and University of Amsterdam (1996)
13. Cramer, R., Damgård, I.B., MacKenzie, P.D.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
14. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
15. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.G.: Authentic data publication over the Internet. Journal of Computer Security 11(3), 291–314 (2003)
16. Durfee, G., Franklin, M.: Distribution chain security. In: CCS 2000: Proceedings of the 7th ACM conference on Computer and Communications Security, pp. 63–70. ACM, New York (2000)
17. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
18. Gabow, H.N., Bentley, J.L., Tarjan, R.E.: Scaling and related techniques for geometry problems. In: STOC 1984: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, pp. 135–143. ACM, New York (1984)
19. Goodrich, M., Tamassia, R.: Efficient authenticated dictionaries with skip lists and commutative hashing. Technical Report, Johns Hopkins Information Security Institute (2000)
20. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Super-efficient verification of dynamic outsourced databases. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 407–424. Springer, Heidelberg (2008)
21. Goodrich, M.T., Tamassia, R., Triandopoulos, N., Cohen, R.: Authenticated data structures for graph and geometric searching. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 295–313. Springer, Heidelberg (2003)
22. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
23. Haber, S., Horne, W., Sander, T., Yao, D.: Privacy-preserving verification of aggregate queries on outsourced databases. Technical Report of HP Labs, HPL-2006-128 (2006)
24. Hacigümüş, H., Iyer, B., Mehrotra, S.: Providing database as a service. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, February 26 - March 1 (2002)
25. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. 13(2), 338–355 (1984)
26. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. SIAM J. Discret. Math. 5(4), 596–603 (1992)
27. Knuth, D.E.: The art of computer programming. fundamental algorithms, vol. I. Addison-Wesley, Reading (1973)
28. Kundu, A., Bertino, E.: A new model for secure dissemination of xml content. IEEE Transactions on Systems Man and Cybernetics-Part C-Applications Reviews 38(3), 292–301 (2008)
29. Kundu, A., Bertino, E.: Structural signatures for tree data structures. Proc. VLDB Endow. 1(1), 138–150 (2008)

30. Li, F.I., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, pp. 121–132 (2006)
31. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
32. Mao, W.: Guaranteed correct sharing of integer factorization with off-line shareholders. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 60–71. Springer, Heidelberg (1998)
33. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy, pp. 122–134 (1980)
34. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. In: Proceedings of ISOC Symposium on Network and Distributed Systems Security, NDSS 2004 (2004)
35. Narasimha, M., Tsudik, G.: Authentication of Outsourced Databases Using Signature Aggregation and Chaining. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 420–436. Springer, Heidelberg (2006)
36. Pang, H., Tan, K.: Verifying completeness of relational query answers from online servers. ACM Trans. Inf. Syst. Secur. 11(2), 1–50 (2008)
37. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Authenticated hash tables. In: CCS 2008: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 437–448. ACM, New York (2008)
38. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract). In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
39. Santoro, N., Khatib, R.: Labelling and implicit routing in networks. The Computer Journal 28(1), 5 (1985)
40. Schnyder, W.: Planar graphs and poset dimension. Order, 323–343 (1989)
41. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
42. Tygar, J.D.: Open problems in electronic commerce. In: Proceedings of 18th ACM PODS (1999)
43. Vuillemin, J.: A unifying look at data structures. Commun. ACM 23(4), 229–239 (1980)
44. Yannakakis, M.: The complexity of the partial order dimension problem. SIAM Journal on Algebraic and Discrete Methods 3(3), 351–358 (1982)
45. Yuen, T., Huang, Q., Mu, Y., Susilo, W., Wong, D.S., Yang, G.: Efficient non-interactive range proof. In: Ngo, H.Q. (ed.) COCOON 2009. LNCS, vol. 5609, pp. 138–147. Springer, Heidelberg (2009)

A Proof of CLS Assumption

Lemma 1. *In the generic group model, the probability that an adversary that performs k operations solves the CLS problem is at most $\frac{5(k+2)^2}{q}$.*

Proof: We keep track of two lists, $L = \{(F_i, \lambda_i)\}$ and $L_T = \{(F_{i,T}, \lambda_{i,T})\}$, of internal/external representation pairs. Initially, we choose five random external representations corresponding to the internal representations: 1, A, B, C, D. The lists are updated according to the procedures listed below:

- *Group Action in* $\mathbb{G}$: Given internal representation F_1 and F_2, compute $F' = F_1 + F_2$. If $(F', \lambda) \in L$ respond with λ. Otherwise choose a new value λ, add (F', λ) to L, and respond with λ.
- *Inversion in* $\mathbb{G}$: Given internal representation F, compute $F' = -F$. If $(F', \lambda) \in L$ respond with λ. Otherwise choose a new value λ, add (F', λ) to L, and respond with λ.
- *Group Action/Inversion in* $\mathbb{G}_T$: Do the same process as for $\mathbb{G}$, except use L_T.
- *Bilinear Map:* Given internal representation F_1 and F_2, compute $F' = F_1 * F_2$. If $(F', \lambda) \in L_T$ respond with λ. Otherwise choose a new value λ, add (F', λ) to L_T, and respond with λ.

Eventually, the adversary outputs a pair of external representations σ_1 and σ_2. If σ_i is not an external representation contained in L, then create a new internal representation $F = E_i$ and add F, σ_i to L. Clearly, all internal representations in L (resp L_T) have degree ≤ 1 (resp 2). We proceed by showing that the above lists are consistent, and then bound the probability that B's answer is a solution to CLS. We assign random values to A, B, C, and D. We also assign random values to E_1 and E_2 if necessary. It was shown in [41] that a d-degree nonzero polynomial in a group of order q will evaluate to 0 with probability at most $\frac{d}{q}$ when the variables of the polynomial. A corollary to this observation is that two non-equal polynomials of degree d will evaluate to the same value for a random assignment of variables with probability at most $\frac{1}{q}$.

The first type of inconsistency would be two internal representations F and G in L such that $F \neq G$, but $F(a, b, c, d, e_1, e_2) = G(a, b, c, d, e_1, e_2)$. By the results in [41], this only happens with probability at most $\frac{1}{q}$. And since there are at most $k + 2$ pairs in L, the probability of this happening for any pair is at most $\frac{(k+2)^2}{q}$.

The second type of inconsistency would occur if there is two internal representations F_T and G_T in L_T such that $F_T \neq G_T$ by $F_T(a, b, c, d) = G_T(a, b, c, d)$. Now each polynomial in L_T has degree at most 2, and thus the probability that two specific polynomials cause an inconsistency is at most $\frac{2}{q}$. And since there are at most k pairs in L_T, the probability of this happening for any pair is at most $\frac{2k^2}{q}$.

Finally, we show that it is unlikely that there are two polynomials, F and G, in L such that $F(a, b, c, d, e_1, e_2) * G(a, b, c, d, e_1, e_2) = ab + cd$. This would imply that the answer returned by the adversary does not solve the CLS. All polynomials in L have degree one. Also, there are no two one-degree polynomials over (A, B, C, D, E_1, E_2) such that their product is the polynomial $AB + CD$. Consider the polynomial $F(A, B, C, D, E_1, E_2) * G(A, B, C, D, E_1, E_2) - AB - CD$. By the above, this is not the zero polynomial, and it has at most degree 2. Thus given a random assignment to the variables will produce a value of 0 with at most probability $\frac{2}{q}$. By the union bound, the probability of this happening for any pair is at most $\frac{2(k+2)^2}{q}$.

Thus the probability of any of the three events above happening is at most $\frac{5(k+2)^2}{q}$. □