Contractual Agreement Design for Enforcing Honesty in Cloud Outsourcing

Robert Nix and Murat Kantarcioglu

The University of Texas at Dallas,
500 W Campbell Rd,
Richardson, TX 75080
{rcn062000,muratk}@utdallas.edu

Abstract. To save time and money, businesses and individuals have begun outsourcing their data and computations to cloud computing services. These entities would, however, like to ensure that the queries they request from the cloud services are being computed correctly. In this paper, we use the principles of economics and competition to vastly reduce the complexity of query verification on outsourced data. Instead of building a specialized computation system for verifying the result of a single outsourced query, we rely on a second, non-colluding data outsourcing entity, whose services are required only a miniscule fraction of the time. Using a game theoretic model, we show that given the proper incentive structure, we can effectively deter dishonest behavior on the part of the data outsourcing services with a very small expected cost increase. We then prove that the incentive for an outsourcing service to cheat can be reduced to zero under this structure.

Keywords: game theory, data outsourcing, contracts, query verification.

1 Introduction

As the amount of data that we generate increases, so does the time and effort necessary to process and store the data. With an increase in time and effort comes an increase in monetary cost. To this end, many have turned to outsourcing their data processing to "the cloud." Cloud computing services are offered by many large companies, such as Amazon, IBM, Microsoft, and Google, as well as smaller companies such as Joyent and CSC. For example, Google [5] recently launched the Google BigQuery Service, which is designed for exactly this purpose: outsourced data processing. The distributed nature of these cloud services shortens data processing time significantly. In addition, these cloud services provide a massive amount of data storage.

In a perfect world, these cloud providers would impartially devote all the computation necessary to any task paid for by the subscribers. In such a world, the querying process would look like figure 1 (minus the verifier), where the subscriber outsources the data D to the cloud, sends queries (Q), and the cloud

J. Grossklags and J. Walrand (Eds.): GameSec 2012, LNCS 7638, pp. 296–308, 2012.

does the necessary calculations and returns the result ($Q(D)$). However, a cloud provider is a self-interested entity. Since it is very difficult for the users of the cloud to see the inner workings of the cloud service, a cloud provider could "cut corners," delivering a less accurate or incomplete computation result which would take fewer system resources to compute. This would, of course, save computational resources for the provider, provided the subscriber was unable to tell a false result from a true one. Because of this, query verification, or the assurance of query result correctness, has been identified as one of the major problems in data outsourcing [17].

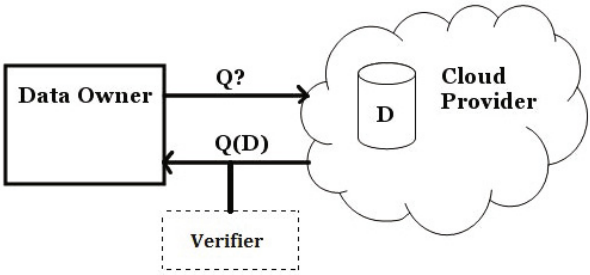

Fig. 1. Data Outsourcing with Verification

Many techniques have been developed and employed for query verification. In figure 1 above, the subscriber sends a query to the outsourcing service, and receives a response. Query verification would then be another process where the subscriber determines if the response is, in fact, the result of the query. The verification process may belong to the owner, or it may be another process entirely. In any case, the verifier aims to make sure that the outsourced server responded correctly. These verification techniques range from simple to extremely complex, and generally rely on the subscriber storing some sketch of the data (much smaller in size), or some cryptographic protocols. Such protocols do a good job verifying the data, but are often slow, or only work with specific types of queries. Many of them assume that the subscriber knows which queries he will execute in advance, so that a sketch can be created for each one. None of these, however, consider the heart of the problem: the self-interest of the parties.

The problem of data outsourcing, and the resultant query verification, is fundamentally a problem of *incentives*. A cloud subscriber wants to get the result of his queries accurately and efficiently, with as low a cost as possible. A cloud provider, however, is most concerned about the profitable use of its computing resources. These incentives can be at odds with each other. The natural way of analyzing competing incentives is to use *game theory*. An interaction between parties is cast as a game, where players use strategy to maximize their gains. The gains from an interaction can be offset artificially by contracts, which can be enforced by law. These adjustments can make actions which were once profitable, such as "cutting corners" in a calculation, less profitable through the use

of penalties. The contracts, therefore, aim not to detect whether a cloud provider is cheating, but to remove the incentive for the provider to cheat altogether.

We propose a game theory-based approach to query verification on outsourced data. We model the process of querying outsourced data as a game, with contracts used to enforce behavior. Data outsourcing does not take place in a vacuum. Service Level Agreements (SLAs) exist for all types of cloud services[12], and are enforceable contracts in court. Thus, we can augment the SLA with an incentive structure to encourage honest behavior. Using a very simple query verification technique, we show that even the threat of verification is enough to deter cheating by a cloud provider.

We consider the case where multiple, non-colluding cloud providers exist. Non-colluding means that the cloud providers do not share information. We believe this is realistic, since cloud providers are competing entities and do not wish to share data with their competitors. In this scenario, we show that without the use of special verification techniques, a data owner can guarantee correct results from rational cloud providers, while incurring an additional cost that is only a small fraction of the overall computation cost.

Our contributions can be summarized as follows:

- We develop a game theoretic model of query verification on outsourced data.
- We show that the model has an equilibrium where the cloud provider behaves honestly.
- Finally, we show that our incentives can improve the expected runtime of *any* query verification method, making it extremely flexible.

Our paper does not consider the privacy of the outsourced data (similar to [2]). However, any privacy-preserving technique for outsourcing data could still be used in our framework. The use of our game theoretic techniques will not affect the privacy-preserving properties of such schemes.

2 Related Work

Several works have outlined query verification methods. The vast majority of these works focus on specific types of queries. Some focus only on selection [1,3,8,18,20], while others focus on relational queries such as selection, projection, and joins [11,10]. Still others focus only on aggregation queries like sum, count, and average [6,19,21]. Some of these processes [16,21] require different verification schemes for each type of query, or even each individual query, requiring that the subscriber knows which queries will be asked in advance.

Many of the aforementioned schemes require complex cryptographic protocols. Some encrypt the data itself, relying on homomorphic schemes to allow the cloud provider to perform the computation [4,19]. A homomorphic operation will always be less efficient than the operation on the unencrypted data, rendering the overhead of these protocols greater by orders of magnitude. Others, such as [16], rely on relatively simpler cryptographic primitives, like secure hash functions. To maintain integrity, our scheme will also use hash functions. Our verification

framework is, however, simpler than these cryptography-based protocols, and can be used to improve the expected runtime of any of these verification schemes.

The work of Canetti, Riva, and Rothblum [2] also makes use of multiple outsourcing services for query verification. However, they make use of all the services all the time, and require a logarithmic number of rounds to ensure verifiability of computation. In addition, they assume that at least one of the cloud providers is in fact honest. We, in contrast, do not assume that any provider is honest, merely that they are *rational* (meaning that the provider wishes to maximize his profits), and we only use additional providers a fraction of the time. In addition, we only require one round of computation.

3 Cryptographic Background

In order to maintain the integrity of our outsourced data, we will need to employ some basic cryptographic primitives. We will need to employ a scheme that allows the owner to make sure that tuples he receives from the server are legitimate, and were not added or altered by the server. We can use a simple message authentication code protocol known as HMAC [13] (Hash-based Message Authentication Code) to do this. HMAC requires the use of *cryptographic hash functions*.

A *cryptographic hash function* or *one-way hash function* is a function mapping a large, potentially infinite, domain to a finite range. This function is simple to compute (taking polynomial time), but is difficult to invert. Equivalently, we can say that, for a cryptographic hash function f, it is difficult to find an x and y such that $x \neq y$ and $f(x) = f(y)$. Examples of cryptographic hash functions include MD5 [15], SHA-1, and SHA-256 [9].

The HMAC system creates a *keyed* hash function from an existing cryptographic hash function. Let m be the message for which we wish to create a code, and k be the key we wish to use. Let f be our cryptographic hash function, and let its required input size be n. If k has a length smaller than n, we pad k with zeroes until it has size n. If k is larger, we let k be $f(k)$ for the purposes of calculating the HMAC function. We define the HMAC function as follows:

$$HMAC(m, k) = f((k \oplus outpad)||f(k \oplus inpad)||m)$$

where outpad and inpad are two constants which are the length of f's block size (in practice, 0x5c...5c and 0x36...36, respectively).

Given a message m and its HMAC value h, if we have the key k, we can simply check to see if HMAC(m,k) matches h. If it does, then the probability that the message is not legitimate (i.e., fabricated or altered) is negligible. Someone who does not have the key k, however, will be unable to compute HMAC(m,k), and will therefore be unable to forge a correct message.

Some more sophisticated methods of verifying data exist, such as Merkle hash trees[7], which allow larger and smaller granularities of the message to be authenticated without authenticating the rest. These other methods of verification could be used to ensure data integrity if desired. In practice, any method of

ensuring data integrity once it is in the hands of the outsourced servers will suffice. We will use the simple HMAC protocol to do this. Data integrity will be a critical component of our second solution.

4 The First Solution

We consider the case where multiple non-colluding cloud providers exist. This means that the parties do not exchange strategies and do not exchange information. Since multiple providers exist, our strategy will be to choose two of them, checking the results of one against the other. We model the query verification process as a game. The game has the following characteristics:

Players (3): the Data Owner(O), and two outsourced servers (S_1 and S_2).

Actions: The data owner begins the game by selecting a probability α, and declares this probability to the servers. He then sends the query (Q) to one of the two servers, with equal probability. With probability α he also sends the query to the other server. If server S_i receives the query, they then respond to the query with either $Q(D)$, that is, the query result on the database D, or $Q'_i(D)$ which is some result other than $Q(D)$. We apply the subscript i to Q' to indicate that one player's method of cheating is different from the other players' method of cheating. We denote the honest action as h, and the cheating action as c. These actions are depicted in figure 4.

Information: Data Owner O has given his database D to S_1 and S_2, with an HMAC message authentication code appended to each tuple. Any message authentication scheme would work here, but its purpose and only effect is that it maintains the integrity of the data. This means that the servers cannot alter any tuples and cannot add any tuples without being detected. The players have entered into an agreement (a contract) before the game, and the contents of this contract are known to all players. The contract could contain the probability α. We assume that no updates are to be made to the database once they are outsourced (they are outsourced purely for the purposes of querying).

Payoffs: The owner recieves the information value of the results received, given by $I_v(Q)$, where Q is either $Q(D)$ or $Q'_i(D)$, minus the amount paid to the servers $P(Q)$. The servers recieve this payment, minus the cost of computing the query, $C(Q)$. For simplicity's sake, we assume that both outsourcing services have the same cost of computation and receive the same payment for the query. The logic below easily applies to the case where costs are different, but this assumption simplifies the equations involved. These payoffs are additionally adjusted by the aforementioned contract. We assume the reservation utility of all parties is zero, and if any party declines the contract, then none of the parties participate.

We assume that $I_v(Q(D)) \geq (1+\alpha)P(Q)$ and $P(Q) \geq C(Q)$. If this were not the case, then the game would not be individually rational without some outside subsidies (that is, some player's expected payout would be less than zero). In essence, we want to ensure that the data owner would want to pay $(1+\alpha)P(Q)$ to receive the result, and the cloud provider would accept $P(Q)$ for the computation. To do this, we make sure that the value that the data owner

places on the query is at least the expected payment, and the cost to the cloud providers is no more than the amount they would be paid. No one takes a loss on the transaction.

We now present two contracts, both of which provide simple solutions to the above game in which neither server has incentive to cheat. The first is very simple and requires no additional computation. The second is intuitively more fair, and thus more likely to be accepted in a real world scenario. Both contracts, however, would be accepted by rational players. It should be noted that both of these contracts are loosely based on the results from Auditing Game II and III in [14].

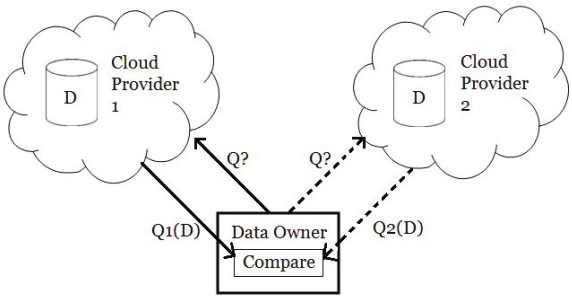

Fig. 2. The Two-Cloud Query Verification System

Contract 1. If the owner asks for query responses from both servers, and the results do not match, both servers pay a penalty of F to the owner, and return the money paid for the computation $P(Q)$ as well.

Theorem 4. The above game with contract 1 has an individually rational, incentive compatible equilibrium in which the servers behave honestly.

Proof: Let $C(Q'_i)$ be the cost of computing Q'_i for S_i. Note that, because S_1 and S_2 do not collude, S_1 does not know Q'_2, and S_2 does not know Q'_1. The only function both know for sure is Q. Without additional knowledge, we can assume that the probability that $Q'_1(D) = Q'_2(D)$ is negligible. For a player to even consider returning Q'_i instead of Q, we must have $C(Q'_i) \leq C(Q)$, since a player will not cheat if they do not gain anything from it. We also assume that $I_v(Q'_i(D)) < 0 < I_v(Q(D))$, since not only is the false result not what the owner asked for, but also appears to be the true result if not verified. If the wrong answer is believed to be correct, this would lead to wrong decisions, and ultimately, financial loss, on the part of the owner. Now, we can define the expected payoffs to each player, where $u_P(x, y)$ is the expected utility for player P when S_1 takes action x and S_2 takes action y. Note that, in these equations and throughout the rest of the paper, we omit the argument D from Q, since D is fixed. We begin with O. If both players are honest (equation 1), O recieves the value of the information gained from the query, minus the expected payment for the calculation, $1 + \alpha$ times $P(Q)$. If one player is dishonest (equations 2 and 3), then with probability α, O detects this and gets both the honest and

the dishonest result and the fine F from both players. With probability $1-\alpha$, he does not detect this, and gets either the correct value or the incorrect value with equal probability. In the event that both players cheat (equation 4), they are once again caught with probability α, but in this case, when they are not caught, O receives only bogus values. This results in the following equations:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q) \tag{1}$$

$$u_O(h,c) = \alpha(2F + I_v(Q) + I_v(Q'_2)) + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_2)) - P(Q)) \tag{2}$$

$$u_O(c,h) = \alpha(2F + I_v(Q) + I_v(Q'_1)) + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_1)) - P(Q)) \tag{3}$$

$$u_O(c,c) = \alpha(2F + I_v(Q'_1) + I_v(Q'_2)) + (1-\alpha)(\frac{1}{2}(I_v(Q'_1) + I_v(Q'_2)) - P(Q)) \tag{4}$$

For the servers, if both servers are honest (equations 5 and 8), they receive the payment for the query, minus the cost of the query, provided they are selected to perform the calculation. This selection probability is why the equations below contain $\frac{1}{2}$. Otherwise, they gain nothing and lose nothing. If one player is dishonest, that player (equations 7 and 10), regardless of whether the other player is honest, with probability α is caught, and loses the fine F. With probability $1-\alpha$, the player is not caught, and gains the payment $P(Q)$, minus the cost of computing his cheat, $C(Q'_i)$, if he is chosen for the computation. If a player is honest while the other player is dishonest (equations 6 and 9), that player similarly is punished with probability α, but invests a cost of $C(Q)$ instead of $C(Q'_i)$ in the computation. This gives us the following equations:

$$u_{S_1}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{5}$$

$$u_{S_1}(h,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) - \alpha F \tag{6}$$

$$u_{S_1}(c,h) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q'_1)) - \alpha F \tag{7}$$

$$u_{S_2}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{8}$$

$$u_{S_2}(c,h) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) - \alpha F \tag{9}$$

$$u_{S_2}(h,c) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q'_2)) - \alpha F \tag{10}$$

We can now find the α for which the expected value for S_1 is less when he cheats than when he is honest, assuming S_2 is honest. By symmetry, this will be the same for S_2. Thus, we set:

$$\frac{1}{2}(1-\alpha)(P(Q)-C(Q_1'))-\alpha F \leq \frac{1}{2}(1+\alpha)(P(Q)-C(Q))$$

Let H represent the quantity $P(Q)-C(Q)$, and H' represent the quantity $P(Q)-C(Q_1')$. Distribute the $(1+\alpha)$ and $(1-\alpha)$ to get:

$$\frac{1}{2}(H')-\frac{\alpha}{2}(H')-\alpha F \leq \frac{1}{2}(H)+\frac{\alpha}{2}(H)$$

Rearranging and combining terms, we get:

$$\frac{1}{2}(C(Q)-C(Q_1')) \leq \alpha F+\alpha P(Q) + \frac{\alpha}{2}(C(Q)-C(Q_1'))$$

Let G represent the quantity $C(Q)-C(Q_1')$, that is, the amount the server would gain from cheating. Substituting this in and factoring out an α, we get:

$$\frac{1}{2}G \leq \alpha(F+P(Q)+\frac{1}{2}G)$$

Multiplying through by two, we get:

$$G \leq \alpha(2F+2P(Q)+G)$$

And, solving for α,

$$\frac{G}{2F+2P(Q)+G} \leq \alpha \tag{11}$$

Since we can define F to be whatever we want in the contract, we can make this minimum α value arbitrarily small. If α is at least this much, then S_1 (and by symmetry, S_2) has no incentive to cheat. If S_2 is not honest, then S_1 has no incentive to be honest, but the payout is less for both (much less, if F is large). Therefore, the best outcome is for both players to behave honestly.

Now, we need to show that choosing α is incentive compatible for O. Given that both players are honest, O's utility is given as:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q)$$

which, by our assumption, is greater than or equal to zero. Thus, it is individually rational for O. If α is increased, it merely decreases this value, so O has no incentive to increase α. If we decrease α, then S_1 and S_2 will see cheating as the more profitable choice, and will begin cheating. This leads to:

$$u_O(c,c) = \alpha(2F+I_v(Q_1')+I_v(Q_2')) + (1-\alpha)(\frac{1}{2}(I_v(Q_1')+I_v(Q_2'))-P(Q))$$

Now, since our α is less than our prescribed value in equation (11), F is bounded above by $\frac{G}{\alpha} - 2P(Q) - G$. Because of this, as α approaches zero, the first term of the above equation decreases. The second term is negative (as $I_v(Q'_1)$ and $I_v(Q'_2)$ are less than zero), and gets worse as α approaches zero. Thus, if α is less than our prescribed value, O expects to lose value from cheating. So, O has no incentive to deviate from $\alpha = \frac{G}{2F+2P(Q)+G}$.

Now, in practice, O does not know G. Thus, he must choose the smallest α that he knows he can use. Since $P(Q) \geq C(Q) \geq G$, O can choose $\alpha = \frac{P(Q)}{2F+2P(Q)-P(Q)} = \frac{P(Q)}{2F-P(Q)}$.

Now, based on the above analysis, it is clear that a cheater will gain less than an honest player when the value of α is chosen as above, regardless of whether the other player is honest. Thus, S_1 and S_2 have no incentive to cheat, and this is incentive compatible for these players as well.

Now, quickly, a note on individual rationality: O has expected payout of $Iv(Q) - (1+\alpha)(P(Q)$. If this is greater than the reservation utility (zero), then the contract is individually rational for O. By our initial assumption about the value of the query, this is true. S_1 and S_2, in equilibrium, have an expected payout of $\frac{1}{2}(1+\alpha)(P(Q) - C(Q))$. Again, by the above assumption, this is true.

As this is both incentive compatible and individually rational for all players, this contract creates the best possible equilibrium where S_1 and S_2 do not cheat, and O pays only $(1+\alpha)$ times the price of a single computation (where α is small). □

5 A More Intuitively Fair Solution

Now, it might seem unfair to punish both players when only one player cheats. The rational player would see the above contract as completely fair, but humans are not always completely rational. Thus, we also examine a contract which identifies the cheater and punishes only the cheater.

Contract 2. If the owner asks for query responses from both servers, and the results do not match, the owner performs a potentially costly audit of the computation. Each server whose result does not match the result given by the owner's process pays a fine F to the owner.

Theorem 5. The above game under contract 2 also has an equilibrium where both servers remain honest.

Proof: Let all variables be defined as above, and let $c(A(Q, Q'_1, Q'_2))$ represent the cost of auditing the computation. The payout functions associated with this contract are as follows:

We begin with O. If both players are honest (equation 12), O recieves the value of the information gained from the query, minus the expected payment for the calculation, $1 + \alpha$ times $P(Q)$. If one player is dishonest (equations 13 and 14), then with probability α, O detects this and gets both the honest and the dishonest result and the fine F from the dishonest player. In this case, he also pays for a costly audit ($c(A(Q, Q'_1, Q'_2))$ to determine which player cheated. With

probability $1-\alpha$, he does not detect this, and gets either the correct value or the incorrect value with equal probability. In the event that both players cheat (equation 15), they are once again caught with probability α, and both pay the fine. However, O only recieves false values, and still pays for the audit. This results in the following equations:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q) \tag{12}$$

$$u_O(h,c) = \alpha(F + I_v(Q) + I_v(Q'_2) - c(A(Q,Q'_1,Q'_2))) + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_2)) - P(Q)) \tag{13}$$

$$u_O(c,h) = \alpha(F + I_v(Q) + I_v(Q'_1) - c(A(Q,Q'_1,Q'_2))) + (1-\alpha)(\frac{1}{2}(I_v(Q) + I_v(Q'_1)) - P(Q)) \tag{14}$$

$$u_O(c,c) = \alpha(2F + I_v(Q'_1) + I_v(Q'_2) - c(A(Q,Q'_1,Q'_2))) + (1-\alpha)(\frac{1}{2}(I_v(Q'_1) + I_v(Q'_2)) - P(Q)) \tag{15}$$

For the servers, if both servers are honest (equations 16 and 19), they receive the payment for the query, minus the cost of the query, provided they are selected to perform the calculation. This selection probability is why the equations below contain $\frac{1}{2}$. Otherwise, they gain nothing and lose nothing. If one player is dishonest, that player (equations 18 and 21), regardless of whether the other player is honest, with probability α is caught, and loses the fine F. With probability $1-\alpha$, the player is not caught, and gains the payment $P(Q)$, minus the cost of computing his cheat, $C(Q'_i)$, if he is chosen for the computation. In this case, if a player is honest while the other player is dishonest (equations 17 and 20), the player is not punished, and therefore receives exactly the same payment as if both players were honest. This gives us the following equations:

$$u_{S_1}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{16}$$

$$u_{S_1}(h,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) \tag{17}$$

$$u_{S_1}(c,h) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q'_1)) - \alpha F \tag{18}$$

$$u_{S_2}(h,h) = \frac{1}{2}(1+\alpha)(P(Q) - C(Q)) \tag{19}$$

$$u_{S_2}(c,h) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q)) \tag{20}$$

$$u_{S_2}(h,c) = u_{S_1}(c,c) = \frac{1}{2}(1-\alpha)(P(Q) - C(Q'_2)) - \alpha F \tag{21}$$

We can now find the α for which the expected value for S_1 is less when he cheats than when he is honest, assuming S_2 is honest. By symmetry, this will be the same for S_2. Thus, we set:

$$\frac{1}{2}(1-\alpha)(P(Q)-C(Q'_1))-\alpha F \leq \frac{1}{2}(1+\alpha)(P(Q)-C(Q))$$

This inequality is exactly the same as in theorem 4. Thus, letting G represent the quantity $C(Q)-C(Q'_1)$, we get:

$$\frac{G}{2F+2P(Q)+G} \leq \alpha \tag{22}$$

Since we can define F to be whatever we want in the contract, we can make this minimum α value arbitrarily small. If α is at least this much, then S_1 (and by symmetry, S_2) has no incentive to cheat. If S_2 is not honest, then S_1 has no incentive to be honest, but the payout is less for both (much less, if F is large). Therefore, the best outcome is for both players to behave honestly.

Now, we need to show that choosing α is incentive compatible for O. Given that both players are honest, O's utility is given as:

$$u_O(h,h) = I_v(Q(D)) - (1+\alpha)P(Q)$$

which, by our assumption, is greater than or equal to zero. Thus, it is individually rational for O. If α is increased, it merely decreases this value, so O has no incentive to increase α. If we decrease α, then S_1 and S_2 will see cheating as the more profitable choice, and will begin cheating. This leads to:

$$\begin{aligned} u_O(c,c) = \alpha(2F + I_v(Q'_1) + I_v(Q'_2) - c(A(Q,Q'_1,Q'_2))) \\ + (1-\alpha)(\frac{1}{2}(I_v(Q'_1)+I_v(Q'_2)) - P(Q)) \end{aligned}$$

As in theorem 4, the first term of this equation decreases as α tends to zero (regardless of $c(A(Q,Q'_1,Q'_2))$, and the second term is negative. Thus, as α decreases, O's expected payout decreases. Therefore, O has no incentive to adjust α up or down, and this α is incentive compatible for O. The arguments in theorem 4 for the incentive compatibility of the servers and the individual rationality of both players continue to apply in this case. Thus, the contract is both incentive compatibile and individually rational for all parties involved. □

The audit process mentioned above could be done in several ways. The simplest, although most expensive, of these would be for the owner to retrieve all the data, then perform the query himself. Obviously, this defeats the purpose of data outsourcing. Based on the fact that the outsourced data uses some message authentication codes to keep the data from being modified, we can improve this. First, for selection queries, if one player fails any MAC checks, then they are obviously cheating. If one player returns fewer results than the other, then they are also obviously cheating. For aggregate queries, we can have each source return the tuples which were selected for the aggregation process. We can then check to see if the aggregate query result matches the values returned by the server. Finding a tuple set that matches a false query result might prove incredibly difficult if the false query was not generated from a sample. We can also apply the

same techniques used for selection queries, noting that the cloud that returns fewer tuples must be cheating (provided all tuples returned are authenticated). Essentially, for a given query, we end up asking the providers to "show their work," or face consequences.

Note the generality of this result. In contrast with many other results, it works for **any query** on **any database** (with the caveat that the query is deterministic), and it works in only one round of computation.

5.1 Conclusions

In summary, by thinking about the problem of query verification from a different perspective, namely, that of an economist, we can drastically reduce the computation required to ensure that the result asked for is the result received. Using the game-theoretic framework outlined here, we show that using a multiple servers, contracts can be designed that will ensure that results obtained from an outsourced computation service are genuine, while requiring only a fractional increase in cost. This is, of course, in contrast to most methods of query verification, which rely on complicated security technologies. The various query verification technologies that are out there are still quite useful, however. Specialized verification methods which take up very little space work well for common queries. They are, however, not generic and can rely on some expensive operations. The outside-the-box approach of using a redundant data service for verification vastly simplifies this process, and incurs a minimal cost.

5.2 Future Work

In the future, we will consider a similar auditing mechanism using only a single cloud service. This mechanism will use both a costly full audit and a less costly partial audit to achieve minimal cost. We also will consider the use of other verification methods in a framework such as ours, and how they can be improved through the use of incentives. Finally, we also wish to consider the effect of accidental errors. The steepness of the penalties involved in this project could lead more risk-averse players to balk at the contract. Nevertheless, a rational, risk-neutral player will have no problems with these contracts, and will be incentivized to check for errors before reporting results.

Acknowledgements. This work was partially supported by Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265, National Institutes of Health Grant 1R01LM009989, National Science Foundation (NSF) Grant Career-0845803, and NSF Grant 0964350.

References

1. Atallah, M., Cho, Y., Kundu, A.: Efficient data authentication in an environment of untrusted third-party distributors. In: IEEE 24th International Conference on Data Engineering, pp. 696–704. IEEE (2008)

2. Canetti, R., Riva, B., Rothblum, G.: Practical delegation of computation using multiple servers. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 445–454. ACM (2011)
3. 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)
4. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
5. Google. Google bigquery service (2011)
6. Haber, S., Horne, W., Sander, T., Yao, D.: Privacy-preserving verification of aggregate queries on outsourced databases. Technical report, Citeseer (2006)
7. Merkle, R.: Secrecy, authentication and public key systems (1979)
8. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. ACM Transactions on Storage (TOS) 2(2), 107–138 (2006)
9. National Institute of Standards and Technology. FIPS 180-2, secure hash standard, federal information processing standard (FIPS), publication 180-2. Technical report, Department of Commerce (August 2002)
10. Pang, H., Jain, A., Ramamritham, K., Tan, K.: Verifying completeness of relational query results in data publishing. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 407–418. ACM (2005)
11. Pang, H., Zhang, J., Mouratidis, K.: Scalable verification for outsourced dynamic databases. Proceedings of the VLDB Endowment 2(1), 802–813 (2009)
12. Patel, P., Ranabahu, A., Sheth, A.: Service level agreement in cloud computing. In: Cloud Workshops at OOPSLA (2009)
13. F. Pub. 198, the keyed-hash message authentication code (hmac). Federal Information Processing Standards Publication, 198 (2002)
14. Rasmusen, E.: Games and information: An introduction to game theory. Wileyblackwell (2007)
15. Rivest, R.: The md5 message-digest algorithm (1992)
16. Sion, R.: Query execution assurance for outsourced databases. In: Proceedings of the 31st International Conference on Very Large Databases, pp. 601–612. VLDB Endowment (2005)
17. Sion, R.: Secure data outsourcing. In: Proceedings of the 33rd International Conference on Very large Databases, pp. 1431–1432. VLDB Endowment (2007)
18. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: Proceedings of the 33rd International Conference on Very Large Databases, pp. 782–793. VLDB Endowment (2007)
19. Xu, J., Chang, E.: Authenticating aggregate range queries over multidimensional dataset. Technical report, Cryptology ePrint Archive, Report 2010/050 (2010)
20. Yang, Y., Papadias, D., Papadopoulos, S., Kalnis, P.: Authenticated join processing in outsourced databases. In: Proceedings of the 35th SIGMOD International Conference on Management of Data, pp. 5–18. ACM (2009)
21. Yi, K., Li, F., Cormode, G., Hadjieleftheriou, M., Kollios, G., Srivastava, D.: Small synopses for group-by query verification on outsourced data streams. ACM Transactions on Database Systems (TODS) 34(3), 1–42 (2009)