To Cheat or Not to Cheat - A Game-Theoretic Analysis of Outsourced Computation Verification

Balázs Pejó
University of Luxembourg
2, avenue de l'Université
Esch/Alzette, Luxembourg
balazs.pejo@uni.lu

Qiang Tang
Luxembourg Institute of Science and Technology
5, Avenue des Hauts-Fourneaux
Esch/Alzette, Luxembourg
tonyrhul@gmail.com

ABSTRACT

In the cloud computing era, in order to avoid computational burdens, many organizations tend to outsource their computations to third-party cloud servers. In order to protect service quality, the integrity of computation results need to be guaranteed. In this paper, we develop a game theoretic framework which helps the outsourcer to maximize its payoff while ensuring the desired level of integrity for the outsourced computation. We define two Stackelberg games and analyze the optimal setting's sensitivity for the parameters of the model.

Keywords

Cloud Computing, Computation Verification, Game Theory, Result Integrity

1. INTRODUCTION

In today's data-centric world, all the companies collect as much data as possible from their customers in order to provide better services, e.g. in a form of personalization. However, processing the collected data is often very computation-intensive, making it infeasible for companies without the necessary resources. In the cloud computing era, a natural solution is to outsource these computations to a cloud service provider. In such a case, two issues arise. One is about the integrity of the computed results. The cloud server may provide some fake results instead of spending its own resources in computing the correct ones. Motivations behind such misbehavior could differ, but saving its own cost and deliberately disrupting the service are two obvious ones that we foresee. The other issue is confidentiality, or more generally privacy: the cloud server learns the input as well as the output of the algorithm it runs. As a standard practice, we refer to the outsourcer as *client* and the outsourcee as *server*.

In this paper we focus on the integrity of the results which can be guaranteed using economic approaches to create incentives for the honest behavior. Specifically, the *not-cheating* behavior can be enforced by setting appropriately the payment (for the computation), the fine (for dishonesty) and the verification rate. For non-critical applications, such as recommendation systems the client might even tolerate some number of incorrect results.

SCC'17, April 02 2017, Abu Dhabi, United Arab Emirates

DOI: http://dx.doi.org/10.1145/3055259.3055262

1.1 Related Work

Data-mining-as-a-service (DMaaS) [Liu et al. 2013] has been proposed to enable clients with insufficient computing resources to mine large volumes of data through outsourcing to cloud servers. Due to the recently gained popularity of outsourcing, serious security challenges emerged such as confidentiality and integrity [Wong et al. 2007]. Result integrity is usually achieved by some kind of auditing/verification mechanism. For integrity verification, [Wong et al. 2009] proposed a solution leveraging on artificial items. In [Dong et al. 2013], the outsourced results are verified by constructing a set of items from real ones and using these as evidence to check mining results.

Another approach was introduced by [Monrose et al. 1999], where the verification scheme requires partial re-execution of the task. In [Canetti et al. 2011], the authors utilized redundancy over multiple agents where at least one is known to be honest. In [Vaidya et al. 2014] a game theoretic framework was developed to improve the existing verification mechanisms. Since our work is inspired by that, we mention its model in Section 2.4. Another related work is [Tang et al. 2016], where the authors compared the verification mechanisms from [Vaidya et al. 2014] by experimenting on real world datasets, using one and more cloud servers.

In [Pham et al. 2014] the authors researched the outsourced verification computation from an economics aspect and developed a model for determining the optimal price of a contract for outsourcing. They analyzed the one and multiple server settings. In the follow up work [Khouzani et al. 2014], they went one step further and identified the optimal settings for the multi-server case when collusion is allowed. Note, that these works are the closest to our work. Still, this paper is different in several aspects (e.g. in [Pham et al. 2014] the desired level of honesty is 100% while our work tolerates cheating up to $0 \leq \theta \leq 1$ part of the results). A more comprehensive comparison is provided in Section 2.3.

1.2 Our Contribution

Our contribution can be summarized as follows.

- We formulate a new game theoretic model where the client wants to outsource some computation to the server and verify the results. We propose a new payoff matrix and define two Stackelberg games where the role of the leader is interchanged amongst the players.
- We provide a strategy for the leader to maximize its payoff while ensuring that the cheating level is below the tolerated threshold.
- We perform a sensitivity analysis by comparing the optimal settings for different parameters.
- We formulate another game where the client may not verify and find its Nash Equilibriums.

1.3 Organization

The paper is organized as follows. Section 2 introduces the problem and the parameters of the model and compare this work with two similar ones. Section 3 introduces a new game theoretic framework while Section 4 defines the two corresponding games. In Section 5 the sensitivity analysis is performed on the parameters of the model. Section 6 contains another game and the corresponding equilibriums when the verification rate may be zero. The conclusion and future works are in Section 7.

2. PROBLEM STATEMENT

In this section we introduce the problem and the corresponding general setup. The models from [Vaidya et al. 2014] and [Pham et al. 2014] are reviewed, and the corresponding problems with them are highlighted.

The problem is the following: the client wants to obtain the results of a computationally intense computation, so it outsources that to a cloud service provider. The server computes the results where some percent of the final results may be wrong, e.g. it may cheat. The server receives its payment in two chunks: it gets some before returning any results as a prepayment[1] and the rest only after the client do not find any fake result during its computation verification process. Furthermore, if the verification indicates high level of cheating, a punishment is enforced on the server.

2.1 Settings

Following [Pham et al. 2014], we assume the server is *lazy but not malicious*, e.g. its only gain from dishonest computation is its saved cost. This is indeed the case if the server has no business conflict with the client. Both players are rational, expected payoff maximizers. The payoff of the players are linear functions of their costs and payments/benefits. The client's goal is to enforce the desired level of honest computation, which corresponds to tolerated cheating rate θ.

We also assume that an outsourced computation produces $1 < K \in \mathbb{Z}$ outputs, which can be checked (verified) by the client individually. For example, in the case of recommender algorithms [Su and Khoshgoftaar 2009], the K outputs are the prediction results for the end users. Based on the findings in [Tang et al. 2016], without loss of generality, we consider the following cheating and verification strategies.

- *Cheating strategy*: The server sets ρ percent of the K results to be random numbers. The parameter ρ is referred to as the cheating rate.
- *Verification strategy*: The client chooses $0 \leq \sigma$ percent of the K outputs to verify, by recomputing them. The parameter σ is referred to as the verification rate.

We further assume, that a result which is not computed (e.g. guessed) is correct with arbitrarily small probability. Indeed, for any $\epsilon > 0$ this can be achieved by enlarging the output range[2]. As such, we define two detection rates in Equation (1): P_0 for no detection (all the verified results are correct) and P_1 for full detection (there is no correct result in the verified results). Note that $\binom{N}{M} = 0$ if $M > N$. Referring to Equation (1), the cheating rate ρ is unknown to the client, while the verification rate σ is unknown to the server.

$$P_0(K,\sigma,\rho) = \frac{\binom{(1-\rho)K}{\sigma K}}{\binom{K}{\sigma K}} \qquad P_1(K,\sigma,\rho) = \frac{\binom{\rho K}{\sigma K}}{\binom{K}{\sigma K}} \tag{1}$$

2.2 Parameters

The parameters used in this paper are summarized in Table 1.

Name	Meaning	Assumption
ρ	Cheating rate	$0 \leq \rho \leq 1$
σ	verification rate	$0 < \sigma < 1$
K	Number of outputs	$0 < K$
$P_0(K,\sigma,\rho)$	Probability of no detection	$0 \leq P_0 \leq 1$
$P_1(K,\sigma,\rho)$	Probability of full detection	$0 \leq P_1 \leq 1$
C	Cost of computation	$C = 1$
ψ	Deposit	$0 \leq \psi \leq 1$
W	Full Payment	$1 < W = 1 + w$
F	Fine	$F > 0$
B	Benefit of the correct results	$W < B = 1 + b$
$V(\sigma)$	Verification cost	$0 \leq V(\sigma) \leq 1$
$\kappa_b(\rho)$	Known benefit reducer	$0 \leq \kappa_b(\rho)$
$\lambda_b(\rho)$	Unknown benefit reducer	$\kappa_b(\rho) \leq \lambda_b(\rho)$
θ	Tolerated cheating rate	$0 \leq \theta \leq 1$

Table 1: The parameters used in this paper

To simplify our discussion, we make some assumptions. First of all the cost for the server to carry out the computations (C) is known to both parties. In practice, this cost can represent the number of operations, the required time for the computation, etc. It is determined by the outsourced algorithm and by the used dataset. We use it as normalizer hence we set $C = 1$.

2.2.1 Transaction variables

The deposit ψ is the amount what the server gets in advance of any calculation. If the cheating is not detected, the client pays the remaining $W - \psi$ amount, so the server's full payment is W. We define $\nu = W - \psi$ and $w = W - C$, so w is the pure gain of the server in case of honest behavior. We call w as fee in the rest of the paper. In case of detection (e.g. some of the verified results are incorrect) no

[1] Called deposit ψ through the paper.

[2] In [Pham et al. 2014] the authors showed such a mechanism.

further payment is done, so as W represents the payment in case of undetected cheating, the deposit ψ can represent the payment in case of detected cheating. In case of full detection so if the client do not find any correct result during its verification[3], the fine F is enforced on the server. If the contract itself is not binding, such a fine can be enforced by introducing a third party as a judge.

2.2.2 Client specific variables

The benefit B for the client is the gain from the correctly computed results. $B > W$, otherwise the client would not outsource at all. We define $b = B - C$ and call it as income in the rest of the paper. The verification cost function $V(\sigma)$ determines the cost corresponding of the amount of verification. It grows monotone in σ. We have $V : [0, 1] \to [0, C]$. By definition $V(0) = 0$ and $V(1) = C$. Both the benefit B and the verification cost function $V(\sigma)$ are determined by the outsourced algorithm and the dataset, so it is common knowledge. If the sum of the verification cost $V(\sigma)$ and the payment W is higher than the client's benefit B, it will not outsource at all because its payoff would be negative. Due to this, B represent a limit (budget) in the total cost $(W + V)$ of the client. The tolerated cheating rate θ is a threshold for cheating: the client wants to obtain the computed results with less than this level of cheating. This is also predetermined by the outsourced algorithm's and/or the data's sensitivity: for example a recommendation system tolerates more cheating than calculating the courses of asteroids traveling towards Earth.

The benefit reducers (i.e. $\lambda_b(\rho)$ and $\kappa_b(\rho)$) are the loss factors for the client if the server cheats. In other words, if the cheating is caught, the benefit[4] for the client is $B - \kappa_b(\rho)$, while if the cheating is undetected it is $B - \lambda_b(\rho)$. They are growing monotonously in the cheating rate ρ. $\lambda_b(0) = \kappa_b(0) = 0$ and $\lambda_b(1) \geq \kappa_b(0) \geq b$, so there is no loss if there is no cheating and the loss in case of 100% cheating is at least the income b. If the unknown loss $\lambda_b(\rho)$ would be smaller than the known loss $\kappa_b(\rho)$ for any cheating rate, the client would prefer not to detect the cheating over detecting it. This is clearly an unrealistic scenario, so to avoid this "ignorance is bliss" situation, we assume the undetected cheating reduce the client's benefit at least as much as the detected one ($\kappa_b(\rho) \leq \lambda_b(\rho)$ for $\forall \rho \in [0, 1]$). Indeed, in case of detection the benefit loss migh be reducible with some form of mitigation (e.g. only use the results in a non-critical environment). These functions are predetermined by the outsourced data/algorithm just like B and V, so the server also knows them.

For simplicity, we discretize the variables using K, so $(\rho, \sigma, \theta, w, \psi) \in \{\frac{i}{K}\}_{i=0}^{K}$.

2.3 Comparison to Khouzani, Cid and Pham

The main topic and the approach to the problem in our paper is similar to [Pham et al. 2014], however, it is fundamentally different in several aspects: in their paper the client wants total honesty, while our approach allows some level of dishonesty defined by tolerated cheating rate θ. Indeed, only mission-critical applications require 100% honesty, and in less critical applications such as recommendation systems this requirement can be relaxed.

Furthermore the authors in [Pham et al. 2014] only considered the cost as a payoff for the client, however, the client's payoff depends on other factors as well such as the benefit B of obtaining the correct result or the losses λ, κ due to the result manipulation.

Another difference is in the number of servers: they mainly focus on the two server - one client case [Khouzani et al. 2014] while we focus on the one server case. Due to this, their client's budget is at least twice the cost of the calculation. In [Pham et al. 2014], there is a result for the one server - one client case, however, their scenario is different from ours: for them punishment was always applicable when cheating was detected, while we only enforce that that in case of Full Detection (see Equation (1)). This means, if the verification find correct and incorrect results as well (e.g. $1 - P_0 - P_1$), the fine is not applicable, rather a deposit ψ is payed[5]. Furthermore, in [Pham et al. 2014] the fine F is a variable which can be maximized to discourage the cheating while we assume it is a a predetermined constant.

Last, there is a difference in limiting some parameters. In [Pham et al. 2014], the probability of auditing (corresponds to our verification rate σ) is limited via the auditing capacity while the reward (corresponds to our payment W) has its own limit (called budget). However, we limit them together using the verification cost function and the benefit: $W + V(\sigma) \leq B$.

2.4 Decision model by Vaidya, Yakut and Basu

In [Vaidya et al. 2014], the authors introduced a game-theoretic approach which can be used on top of a verification mechanism for the outsourcing scenario. The proposed payoff matrix for the server is shown in Table 2 where $X \leq W - C$ is the payoff for the server when cheating is detected.

	cheat	*not-cheat*
Detected	X	$W - C$
Not-detected	$W - (1 - \rho)C$	$W - C$

Table 2: Payoff Matrix in [Vaidya et al. 2014]

The payoff is basically the payment minus the cost of executed calculation. The goal is to prevent the server from cheating by setting the parameters in a way that *not-cheat* is a dominant strategy. Given a detection rate $P_D = 1 - P_0$, they showed that it can be established when Inequality (2) is true.

$$\begin{aligned} W - C \geq & (1 - P_D)(W - (1 - \rho)C) + P_D X \Rightarrow \\ P_D \geq & \frac{\rho C}{W - X - (1 - \rho)C} \end{aligned} \tag{2}$$

[3]This kind of fine enforcement does not make sense if the client verify significant amount of results, however, our analysis shows this is not the case so such a punishment enforcement is feasible.

[4]We do not consider the positive effect of the recalculated results, since the client only verify a very small fraction of the whole as we show in Section 4.

[5]Relaxing the conditions of the fine enforcement only gives more incentives for the server to cheat, however, even with this "help" the server will prefer not to cheat as we will show later.

This is a decision model for the server, since detect/not-detect is not a choice for the client. To make it a game theoretic model, the client has to have choices as well, such as setting the verification rate σ (or set the deposit ψ and the fee w). Furthermore, the client's payoff needs to be defined. Also it would be more appropriate if the payment X in case of detected cheating would be represented by a *cost* and a *payment* just like in the rest of the cases in the payoff matrix.

3. THE NEW GAME THEORETIC MODEL

We define a new payoff matrix and set two conditions which guarantee that the client prefers the honest result while the server will not cheat more than the tolerated cheating rate.

The normal form representation of the game is a tuple $\{N, \Sigma, u\}$, where the players are N=$\{$client, server$\}$, the actions are $\Sigma_c = \{\frac{1}{K}, \ldots, \frac{K-1}{K}\}$ and $\Sigma_s = \{0, \frac{1}{K}, \ldots, 1\}$ which representing the fraction of the K results being verified/cheated and the payoff functions shown in Equation (3).

3.1 Utilities

The payoff matrix is shown in Table 3. Note, that we treat $b, \theta, F, \lambda_b, \kappa_b$ and V as predetermined constants/functions, and only σ and ρ as variables. For each possible cheating/verification rate there is a corresponding fee/deposit pair (w, ψ) which gives the server/client the highest payoff[6].

		earnings	losses
Full Detect	Server	ψ	$-(1-\rho)C - F$
P_1	Client	$B + F$	$-\kappa_b(\rho) - \psi - V(\sigma)$
Detect	Server	ψ	$-(1-\rho)C$
$1 - P_0 - P_1$	Client	B	$-\kappa_b(\rho) - \psi - V(\sigma)$
Not-Detect	Server	W	$-(1-\rho)C$
P_0	Client	B	$-\lambda_b(\rho) - W - V(\sigma)$

Table 3: Our Payoff Matrix

If there is no cheating, the payoff for the client is its own cost (fee w for the server and verification cost $V(\sigma)$) deducted from its income b, while the server's payoff is exactly the fee w. Recall that due to normalization the cost of the computation is $C = 1$. If there is cheating with no detection, the server's payoff grows with the cheating rate ρ, while the client's payoff decreases with the unknown benefit reducer λ. On the other hand, if it is detected, κ is used instead of λ and the remaining payment $W - \psi$ is not payed. Furthermore, in case of Full Detection, a fine F is enforced.

$$\begin{aligned} U_C(\sigma,\rho) =& P_0(b - \lambda_b(\rho) - w - V(\sigma)) + \\ & P_1(1 + b + F - \kappa_b(\rho) - \psi - V(\sigma)) + \\ & (1 - P_0 - P_1)((1 + b - \kappa_b(\rho) - \psi - V(\sigma)) \\ U_S(\sigma,\rho) =& P_0(w + \rho) + \\ & P_1(\psi - F - (1-\rho)) + \\ & (1 - P_0 - P_1)(\psi - (1-\rho)) \end{aligned} \tag{3}$$

[6]More details about the optimal (w, ψ) selection is in the following section.

3.2 Conditions

3.2.1 Client's condition

We want to avoid the case when *cheat* results in more gain for the client than *not-cheat*. To ensure that the honest result provides the highest payoff for the client, we require Inequality (4) to hold where ν is the remaining payment after the deposit, e.g. $\nu = W - \psi$. Note, that $0 \le \nu \le B$.

$$\begin{aligned} U_C(\sigma,0) \ge& U_C(\sigma,\rho) \ \ \forall \rho > 0 \ \ \Rightarrow \\ b - w - V(\sigma) \ge& P_0(K,\sigma,\rho)[b - \lambda_b(\rho) - w - V(\sigma)] + \\ & P_1[1 + b + F - \kappa_b(\rho) - \psi - V(\sigma)] + \\ & (1 - P_0 - P_1)[1 + b - \kappa_b(\rho) - \psi - V(\sigma)] \ \ \Rightarrow \\ \nu = 1 + w - \psi \le& \kappa_b(\rho) + \frac{P_0\lambda_b(\rho) - P_1 F}{1 - P_0} \end{aligned} \tag{4}$$

3.2.2 Server's condition

We want to force the server not to cheat above the client's tolerated cheating rate θ. This can be guaranteed if the payoff in case of cheating with θ is not less that the payoff for any cheating rate above it. This means, Inequality (5) must be satisfied for all $\rho \ge \theta$ where $P_i^j = P_i(K, \sigma, j)$ where $i \in \{0, 1\}$ and $j \in \{\rho, \theta\}$.

$$\begin{aligned} U_S(\sigma,\theta) \ge& U_S(\sigma,\rho) \ \ \forall \rho \ge \theta \ \ \Rightarrow \\ \nu = 1 + w - \psi \ge& \frac{F(P_1^\rho - P_1^\theta) + \theta - \rho}{P_0^\rho - P_0^\theta} \end{aligned} \tag{5}$$

4. THE STACKELBERG GAMES

This section defines the two Stackelberg game and the optimal offer what the leader should decline. An example is provided for both games.

Based on the payoff matrix in Table 3, we defined two Stackelberg games. First the client is the leader and the server is the follower, while in the second the roles are switched: the server is the leader and the client is the follower. Note, that the leader's move is represented via the payment/deposit pair, however, these are actually determined by the verification/cheating rates: for each rate there is a (w, ψ) pair which maximizes the leader's payoff. In other words, the leader chooses its verification/cheating rates, and declares the follower the corresponding payment-deposit pair as its move/play.

In this game, the leader can make an offer (w, ψ)[7] to the follower. If the follower rejects this, the game terminates just as in case when no offer is made, and the payoffs are zero for both players. If the offer is accepted, then the server carries out the computation with some level of cheating. After receiving the results from the server, the client verifies some percentage of it and acts accordingly: (1) either pays the remaining amount in case the verified results are all correct or (2) refuse to pay in case there are some incorrect results or (3) even enforce a fine in case no correct result found. The extensive form of this game is shown in Figure 1.

[7]Technically, the offer is (ν, ψ), so the payment after and before the computation, but this corresponds to the pair $(\nu - 1 + \psi, \psi) = (w, \psi)$.

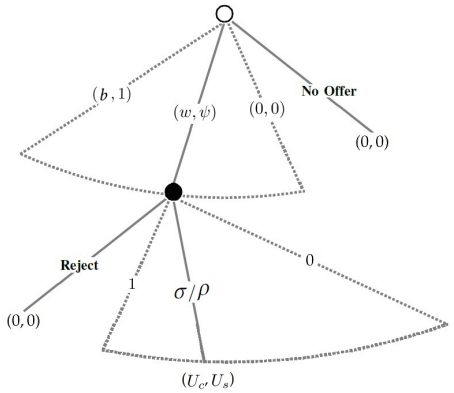

Figure 1: The game's extensive form

4.1 Client is leading

We assume, the client choose σ first. In more detail, for a particular σ, Inequality (4) defines an upper limit y for ν while Inequality (5) defines a lower limit x, i.e. they define an interval $[x, y]$ for the payment after verification ν where both the client's and the server's conditions hold. The client wants to set w and ψ in a way that it maximize its payoff. Since both w and ψ are spending for the client, it would like to minimize them. To determine (w, ψ), the client proceeds as follows where $\hat{1} = 1 + \frac{1}{K}$.

1. If the interval is empty (i.e. $x > y$), there is no value ν which satisfies both Inequality (4) and (5), so no solution exists for the σ chosen in advance by the client.

2. If $[x, y] \cap [0, B] = \varnothing$: ν is by definition $\nu \in [0, B]$, so no solution exists for the chosen σ.

3. If $\hat{1} \in [x, y]$: both variable can be fully minimized, e.g. $(w, \psi) = (\frac{1}{K}, 0)$.

4. If $0 < y < \hat{1}$: This means, the possible values of $w - \psi$ are negative. This case only w can be fully minimized, e.g. $w = \frac{1}{K}$. ψ is minimal if $\nu = y$, which case $\psi = \hat{1} - y$.

5. If $\hat{1} < x < B$: This means, the possible values of $w - \psi$ are positive. This case only ψ can be fully minimized, e.g. $\psi = 0$. w is minimal if $\nu = x$, which case $w = x - 1$.

4.1.1 Example

We set $K = 1\,000\,000$, $b = \frac{3}{4}$, $F = 1$, $\kappa_b(\rho) = (1 + b)2\rho = 3.5\rho$, $\lambda_b(\rho) = (1 + b)4\rho = 7\rho$, $V(\sigma) = \sigma$ and $\theta = 0.1$, the two limits from Inequality (4) and (5) are shown in Figure 2 for $\sigma = \frac{10}{K}$. With this settings, checking 10 values (out of 1 million) results in no possible solution (Case (1): $x > y$). One can check that the only σ's where the corresponding limits are feasible (e.g. $x < y$) shown in Table 4 with the optimal payment/deposit choices.

We require σ to be higher than $\frac{1}{K}$, otherwise the detection and full detection case would be indifferent which means the punishment is always enforced, simplifying our game to the one in [Pham et al. 2014]. Figure 3 shows the server's payoff for the viable verification rates. It is visible, that independently from σ, the server is better of by not cheating (independently from the tolerated cheating rate θ). Due to this, the client will only verify 2 results to maximize its payoff for the case $\rho = 0$. This means $(\sigma, \rho) = (\frac{2}{K}, 0)$ is the Nash Equilibrium where $(w, \psi) = (\frac{1}{K}, 0)$.

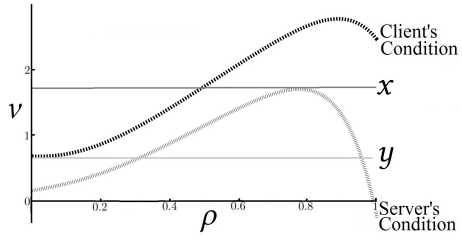

Figure 2: The interval defined by Inequality (4) and (5) for $\sigma = \frac{10}{K}$.

σ	$[x, y]$ for ν	(w, ψ)
$2/K$	$[0.473\,684, 2.366\,600]$	$(\frac{1}{K}, 0)$
$3/K$	$[0.462\,496, 2.089\,780]$	$(\frac{1}{K}, 0)$
$4/K$	$[0.587\,682, 1.667\,837]$	$(\frac{1}{K}, 0)$
$5/K$	$[0.742\,010, 1.358\,537]$	$(\frac{1}{K}, 0)$
$6/K$	$[0.908\,944, 1.142\,593]$	$(\frac{1}{K}, 0)$

Table 4: Feasible solutions for the example

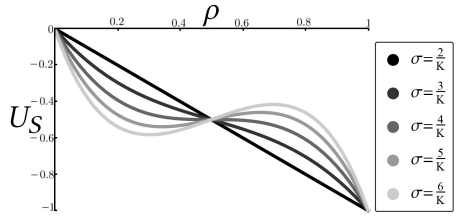

Figure 3: The Server's Payoff when the client is the leader

4.2 Server leading

Recall that all variables are common knowledge, so the server can calculate for any σ^* the corresponding interval derived from Inequality (4) and (5). The question is the same again, but this time the server is asked: how to set (w, ψ) in a way that its payoff function is maximized. Since both the payment and the deposit are earnings for the server, its goal is to maximize both. Note, that the client's payoff must be positive, otherwise it will not accept any offer. For a particular σ the payment w could be at most $b - \frac{1}{K} - V(\sigma)$ otherwise the previous condition would not hold even for $\rho = 0$. We define $\hat{b} = b - \frac{1}{K} - V(\sigma)$.

1. If the interval is empty (i.e. $x > y$), no solution exists for the corresponding σ.

2. If $[0, B] \cap [x, y] = \varnothing$: ν is by definition $\nu \in [0, B]$, so no solution exists for the corresponding σ.

3. If $\hat{b} \in [x, y]$: both variable can be fully maximized, e.g. $(w, \psi) = (\hat{b}, 1)$.

4. If $\hat{b} < x$: This case only w can be maximized fully, e.g. $w = \hat{b}$. ψ is maximal if $\nu = x$, which case $\psi = 1 + \hat{b} - x$.

5. If $y < \hat{b}$: This case only ψ can be maximized fully, e.g. $\psi = 1$. w is maximal if $\nu = y$, which case $w = y$.

The server compares the maximal payments and deposits for the possible verification rates and chooses the maximum which will be the offer for the client. Than the client checks, which σ's corresponding interval contains this particular offer and choses the one which provides the highest payoff.

4.2.1 Example

Following our previous example where $b = 0.75$, using the intervals shown in Table 4, the server's offers with the corresponding verification rates shown in Table 5, where the maximum is $(w, \psi) = (0.749\,997, 1)$. After receiving the offer, the client's possibility in case of acceptance is only to verify $\sigma = \frac{2}{K}$ results, otherwise its payoff would be negative even for $\rho = 0$. The client's payoff is shown in Figure 4 for the offers from Table 5 with the corresponding verification rates.

σ	w	ψ
0.000 002	0.749 997	1
0.000 003	0.749 996	1
0.000 004	0.749 995	1
0.000 005	0.749 994	1
0.000 006	0.749 993	0.841 053

Table 5: Best offers for the server for different verification rates

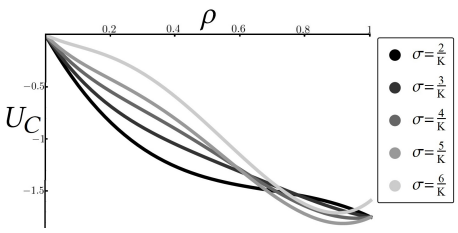

Figure 4: The Client's payoff when the server leads

4.3 Remark

It seems from the two games that the leading player has overwhelming advantage. In the majority of the cases it can reach its maximal payoff ($b - \frac{2}{K}$ for the client and $b - \frac{3}{K}$ for the server), while the follower gets only the smallest viable payoff $\frac{1}{K}$. In practice, the server seems to be the leader by setting the prices for different services. Still, these prices are actually much lower because the cloud servers must compete for the customers.

Note, that the server's payoff can still be positive when $\rho > \theta$, however, a rational server will *not-cheat* due to its incentives[8]: its payoff is maximal when $\rho = 0$. This means, the optimal choice for the client would be to only verify the minimum, e.g. the Nash Equilibrium of the game - if exists - is $(\sigma, \rho) = (\frac{2}{K}, 0)$.

In both game, *not-cheat* is the dominant strategy for the server despite that certain θ level of cheating is allowed. Actually, Inequality (5) ensures, that $U_S(\theta) > U_S(\rho)$ if $\rho > \theta$. The server's payment decreases rapidly at 0 for $\sigma > \frac{2}{K}$, so by having the tolerated cheating rate θ further from 0, the lower $U_S(\theta)$ gets, making the Inequality stronger. One would expect, the higher the toleration rate, the more relaxed the condition. Despite, exact opposite happens.

So far we were only talking about intentional cheating, however, non-intentional cheating (e.g. miscalculations, etc.) may occur too. To give some additional level of protection against such a thing, the client may verify slightly more. By doing so, it decrease its payoff in case of $\rho = 0$ as it is shown in Figure 5, but it also straightens the payoff curve around $\rho = 0$. So for the client it maybe a good idea to verify as much as possible based on Inequality (4) and (5).

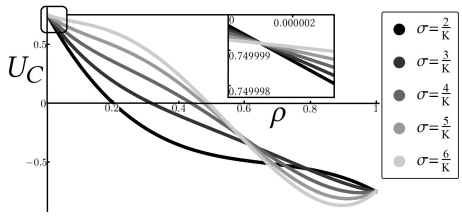

Figure 5: The Client's payoff when (s)he leads

5. SENSITIVITY ANALYSIS

In this section we compare the parameters pairwise to see how much the possible maximum verification rate and the corresponding (w, ψ) changes. We are interested in this - despite that the pure Nash Equilibrium would be almost always to verify only 2 results - because even this maximal verification is insignificant compared to the whole computation, e.g. $< 1\%$. This means, its advantage of smoothening the payoff curve - as it was seen in Figure 4 - overcomes the decreased effect on the payoff when $\rho = 0$. We focus on the first game, however, in the second game similar relations can be found. Our default settings for the parameters are $\theta = 0.1, b = F = C = 1$, and $\lambda_b(\rho) = \kappa_b(\rho) = (1 + b)2\rho$.

With these settings, we found a connection between b and σ: if the income is more, the client can check more (see Table 6, 7 and 8). In the following, we list some cross-comparison results amongst the variables where change appears either in the maximal verification rate allowed by Inequality (4) and (5) or in the optimal value of the payment and/or deposit. The table cells represent σ, w and ψ respectively.

Income - Fine

If we compare the income b with the fine F, we see in Table 6, that for small b and high F no solution exists: the client prefers *cheat* more than *not-cheat* because $P_1 F >> b$.

	$F = 0.5$	$F = 1$	$F = 2$	$F = 4$
$b = 0.2$	$(\frac{4}{K}, \frac{1}{K}, 0.4)$	$(\frac{4}{K}, \frac{1}{K}, 0.4)$	$\perp$	$\perp$
$b = 0.4$	$(\frac{4}{K}, \frac{1}{K}, 0.3)$	$(\frac{4}{K}, \frac{1}{K}, 0.3)$	$(\frac{4}{K}, \frac{1}{K}, 0.3)$	$\perp$
$b = 1.6$	$(\frac{5}{K}, \frac{1}{K}, 0)$	$(\frac{5}{K}, \frac{1}{K}, 0)$	$(\frac{5}{K}, \frac{1}{K}, 0)$	$(\frac{5}{K}, \frac{1}{K}, 0)$

Table 6: Varying b and F

Income - Tolerated Cheating Rate

If we compare σ, w and ψ when b and θ are varied, we see in Table 7 two possible cases. If b is high, high θ is a disadvantage for the client because w must be higher. On the other hand if b is low, high θ is an advantage since ψ decrease.

[8]A *lazy but honest* server's payoff is not affected by the other player loss.

	$\theta = 0.01$	$\theta = 0.1$
$b = 0.1$	$(\frac{4}{K}, \frac{1}{K}, 0.45)$	$(\frac{3}{K}, \frac{1}{K}, 0.266667)$
$b = 1$	$(\frac{6}{K}, \frac{1}{K}, 0.333334)$	$(\frac{5}{K}, \frac{1}{K}, 0.200001)$
$b = 10$	$(\frac{19}{K}, \frac{1}{K}, 0)$	$(\frac{11}{K}, 1.96065, 0)$

Table 7: Varying b and F

Income - Number of Results

Now, lets take a closer look at the number of verifiable results. Surprisingly, more results do not lead to higher σ, only lower w. Furthermore, for higher b higher w corresponds.

	$K = 10^3$	$K = 10^6$	$K = 10^9$
$b = 3$	$(\frac{7}{K}, 1.09119, 0)$	$(\frac{7}{K}, 1.08692, 0)$	$(\frac{7}{K}, 1.08692, 0)$
$b = 5$	$(\frac{8}{K}, 1.28412, 0)$	$(\frac{8}{K}, 1.27803, 0)$	$(\frac{8}{K}, 1.27802, 0)$
$b = 7$	$(\frac{9}{K}, 1.49373, 0)$	$(\frac{9}{K}, 1.48525, 0)$	$(\frac{9}{K}, 1.48524, 0)$

Table 8: Varying b and K

Verification cost

In our example through the paper we set the verification cost function to be linear, however in practice this assumption may not hold, e.g. V is steep closer to zero and moderate around one. By setting $V(\sigma) = \sqrt{\sigma}$, which is a concave function described earlier, one might expect to obtain different results. Therefore we compared $V(\sigma)$ with all other variables, however, the obtained results was the same when $V(\sigma) = \sigma$. This is due to the limited amount of verification: even if $V(\sigma)$ grows rapidly around zero, σ itself is very close to it.

6. EQUILIBRIUMS

This section we define a game where the players actions are reduced, e.g. they only have two options. We analyze it ant define its equilibriums.

We argued in Section 4 that if the client plays *verify*, it should use the highest σ^* amongst all the possible σ's where Inequality (4) and (5) defines a viable interval for ν. This conclusion was only valid when $V(\sigma)$ is linear, however, in Section 5 we showed that this conclusion holds for steeper functions as well. So *verify* reduces to *verify max* which corresponds to σ^*. We also showed that *not cheat* is the dominant strategy for the server for any positive σ. Now, if we allow the client to play *not verify*, the game reduces to 2-2 actions of each player as shown in Table 9 where each cell represents (U_C, U_S).

σ \ ρ	0	1
0	$(b - w, w)$	$(b - \psi - \lambda_b(1), 1 + w)$
σ^*	$(b - w - V(\sigma^*), w)$	$(b - \psi - \kappa_b(1) - V(\sigma^*) + P_1F, \psi - P_1F)$

Table 9: Reduced Payoff Matrix

This game is cyclic, if the following conditions hold.

1. If the client plays *not verify*, the server rather plays *fully cheat*. This is true by definition of the payoff function.
2. If the server plays *fully cheat*, the client prefers to play *verify max*. This holds only if $V(\sigma^*) < \lambda_b(1) - \kappa_b(1) + P_1F$.
3. If the clients plays *verify max*, the server is better of playing *no cheat*. This is correct if $w > \psi - P_1F$.
4. If the server plays *no cheat*, the client will gain more by playing *not verify*. This is satisfied by definition of the payoff function.

If (2) and (3) hold, the game has a unique Mixed Nash Equilibrium [Aumann 1989], with the probabilities defined in Equation (6).

$$\begin{aligned} \Pr(\textit{fully cheat}) &= \frac{1}{1 + \psi - P_1F} \\ \Pr(\textit{verify max}) &= \frac{V(\sigma^*)}{\lambda_b(1) - \kappa_b(1) + P_1F} \end{aligned} \tag{6}$$

If (2) is not true (e.g. $V(\sigma^*) \geq \lambda_b(1) - \kappa_b(1) + P_1F$), the game's Pure Nash Equilibrium is $(0, 1)$, so *not verify - fully cheat*. In this case, the client should not outsource at all, because it cannot verify any results due to the extremely high verification cost. On the other hand, if (2) holds and only (3) is violated (e.g. $w \leq \psi - P_1F$), the Pure Nash Equilibrium is $(\sigma^*, 1)$, which corresponds to *verify max - fully cheat*. This situation is also unacceptable for the client since its payoff would be negative. As a result, the client only participates in the game if (2) and (3) holds along with Inequality (4) and (5), so the only realizable Equilibrium is the mixed one. In this case, the expected payoffs for the players are shown in Equation (7) where $l = \lambda_b(1) - \kappa_b(1)$.

$$\begin{aligned} \mathbb{E}(U_S) &= w + \frac{V(\sigma^*) + \frac{P_1F - (2+w)V(\sigma^*) + l}{1 + \psi - P_1F}}{P_1F + l} \\ \mathbb{E}(U_C) &= b - W - \frac{V(\sigma^*)^2}{P_1F + l} + \frac{W + V(\sigma^*) - \lambda_b(1) - P_1F}{1 + \psi - P_1F} \end{aligned} \tag{7}$$

7. CONCLUSION

Nowadays, outsourcing computation to cloud service providers is very common. Guaranteeing the correctness of these calculations is usually done by verification mechanisms. In this paper, we proposed a game theoretic framework which can be applied on a top of a verification mechanism to maximize the client's payoff while not just ensuring the desired level of correctness but to provide full honesty in case of verification. Furthermore, our results show that the minimal verification is already enough to ensure this.

As a future work, there are numerous ways to extend this model. Making the game sequential or introducing reputation are reasonable future directions. Making the game asymmetric by hiding some variables - for example b from the server - to make the game incomplete information is also interesting idea. Introducing new variables, such as a computational cost difference between the client and the server is a promising direction as well.

Acknowledgments

Qiang Tang is partially supported by a CORE (junior track) grant from the National Research Fund, Luxembourg.

Disclaimer

A previous version of this work was presented as a poster on Conference on Decision and Game Theory for Security (GameSec) in 2016 November 2-4, New York.

8. REFERENCES

[Aumann 1989] Robert J Aumann. 1989. Game theory. In *Game Theory*. Springer, 1–53.

[Canetti et al. 2011] Ran Canetti, Ben Riva, and Guy N Rothblum. 2011. Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 445–454.

[Dong et al. 2013] Boxiang Dong, Ruilin Liu, and Hui Wendy Wang. 2013. Result integrity verification of outsourced frequent itemset mining. In *Data and Applications Security and Privacy XXVII*. Springer, 258–265.

[Khouzani et al. 2014] MHR Khouzani, Viet Pham, and Carlos Cid. 2014. Incentive engineering for outsourced computation in the face of collusion. In *Proceedings of WEIS*.

[Liu et al. 2013] Ruilin Liu, Philippos Mordohai, Wendy Hui Wang, Hui Xiong, Ruilin Liu, Hui Wendy Wang, Philippos Mordohai, and Hui Xiong. 2013. Integrity Verification of K-means Clustering Outsourced to Infrastructure as a Service (IaaS) Providers.. In *SDM*. SIAM, 632–640.

[Monrose et al. 1999] Fabian Monrose, Peter Wyckoff, and Aviel D Rubin. 1999. Distributed Execution with Remote Audit.. In *NDSS*, Vol. 99. 3–5.

[Pham et al. 2014] Viet Pham, MHR Khouzani, and Carlos Cid. 2014. Optimal contracts for outsourced computation. In *International Conference on Decision and Game Theory for Security*. Springer, 79–98.

[Su and Khoshgoftaar 2009] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009), 4.

[Tang et al. 2016] Qiang Tang, Balázs Pejó, and Husen Wang. 2016. Protect both Integrity and Confidentiality in Outsourcing Collaborative Filtering Computations. In *Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on*. IEEE.

[Vaidya et al. 2014] Jaideep Vaidya, Ibrahim Yakut, and Anirban Basu. 2014. Efficient Integrity Verification for Outsourced Collaborative Filtering. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 560–569.

[Wong et al. 2007] Wai Kit Wong, David W Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. 2007. Security in outsourcing of association rule mining. In *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 111–122.

[Wong et al. 2009] Wai Kit Wong, David W Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. 2009. An audit environment for outsourcing of frequent itemset mining. *Proceedings of the VLDB Endowment* 2, 1 (2009), 1162–1173.