Optimal Contracts for Outsourced Computation*

Viet Pham, MHR. Khouzani, Carlos Cid

Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, United Kingdom
{viet.pham.2010, arman.khouzani, carlos.cid}@rhul.ac.uk

Abstract. While expensive cryptographically verifiable computation aims at defeating malicious agents, many civil purposes of outsourced computation tolerate a weaker notion of security, i.e., "lazy-but-honest" contractors. Targeting this type of agents, we develop optimal contracts for outsourcing of computational tasks via appropriate use of rewards, punishments, auditing rate, and "redundancy". Our contracts provably minimize the expense of the outsourcer (principal) while guaranteeing correct computation. Furthermore, we incorporate practical restrictions of the maximum enforceable fine, limited and/or costly auditing, and bounded budget of the outsourcer. By examining the optimal contracts, we provide insights on how resources should be utilized when auditing capacity and enforceability are limited. Finally, we present a light-weight cryptographic implementation of the contracts to mitigate the double moral hazard problem between the principal and the agents.

1 Introduction

The idea of outsourcing complex computation tasks has been proposed and implemented in a variety of applications. Research projects involving complex analysis on a huge multitude of data have utilized parallel processing of their computations on the processors of millions of volunteering Internet users. These include search for extra-terrestrial life (*SETI@Home*), investigation of protein folding and computational drug design (*Folding@Home* and *Rosetta@home*). Businesses from different sections including finance, energy infrastructure, mining and commodities transport, technology and innovation [7] have also realized the benefits of outsourcing their data and computation, and "moving to the cloud". The cloud, as a dedicated infrastructure with specialized man-force and powerful computing capabilities, along with the ability to pool demands from different clients and dynamic assignment of the resources can reduce the cost of computation. Meanwhile, the outsourcer is also relieved from maintaining a dedicated computing infrastructure and in addition, has the total flexibility of pay-per-use paradigm, to flex-on or to flex-off services effortlessly [7]. This growing trend has made possible small virtualised computers and smart devices with powerful computational power, applicable to critical mission scenarios and everyday use.

* The full version of this paper with formal proofs for all the propositions is accessible via https://eprint.iacr.org/2014/374.pdf.

In all of these scenarios, there is a concern for the outsourcer (client) about the correctness of the returned results. The provider of computation services (the servers) have an economic incentive to return guessed results as opposed to performing the computation completely and honestly, and thereby save on the computation work. Hence, to make this paradigm viable and guarantee soundness of the results, there must be an auditing mechanism in place. The auditing, however, is not free: it either creates computational overhead for the client, the server, or both. The auditing can be done by the outsourcer itself or through a trusted third party for a fee, say, through re-computation. Alternatively, a *redundancy* scheme can be employed in which the same job is outsourced to multiple servers and the results are checked against each other.

Irrespective of the auditing mechanism, the outsourcer can set an extremely large fine for detected wrong results, and make cheating theoretically impossible even for the lowest probability of cheat detection. However, in practice, an extremely large fine is a non-credible threat. A more reasonable assumption is a cap on the maximum enforceable fine, with the special interesting case where the cap is zero. In this paper we provide a concrete and general approach based on Principal-Agent modelling from game theory to optimal contract designs for outsourcing from the client (principal) to the servers (agents). Specifically, we assume a general maximum enforceable fine, maximum budget, and costly and/or limited auditing rate. We formulate the utilities of both the principal and the agents, as well as essential constraints that guarantee honest computation (incentive compatibility) along with their acceptance of the offer (participation). This allows us to systematically compute the optimal contract such that the principal's expense is minimized. Our work hence potentially provides a benchmark enabling comparison among different deployments of outsourcing.

The paper is structured as follows: In Section 2, we review previous results and describe our contributions. This is followed by a detailed motivation of our contract model in Section 3, along with descriptions of important constraints that make the problem non-trivial. In Section 4, we compute optimal contracts involving only one agent, and explore related improvements. In Section 5, we allow the principal to also potentially outsource the same task to multiple non-colluding agents as an alternative means of auditing and develop optimal hybrid contracts. We further establish the global optimality of our hybrid two-agent contracts among all possible contracts involving any number of non-colluding agents with respect to the notion of Nash Equilibria. In Section 6, we comment on cryptographic implementation of our contracts, i.e., how to enforce the terms and policies in an automated way. Finally, in Section 7, we conclude the paper with a summary of the results and remark on some potential future directions.

2 Related work

A line of research is focused on designing reliable verification techniques for outsourcing of special-purpose computations. For instance, [17] investigates outsourcing of linear optimizations. Other notable examples are queries on outsourced databases, including typical queries [1, 5] and aggregation [18]. Their

main paradigm is for the querier to rely on trusted information directly given by the data owner (outsourcer) to verify the results returned by the servers.

Verification methods for general-purpose computing also appear in several remarkable works. In [12] verification is performed by re-executing parts of the computation. A variation is presented in [3] in which the authors utilize redundancy over multiple agents, assuming that at least one of them is honest. Outsourced computation has also caught attraction in cryptographic research: in a seminal work, the authors of [8] formally define verifiable computation and give a non-interactive solution. Their solution uses Yao's garbled circuits to represent the computation and homomorphic encryption to hide such circuits from the agents. More efficient but interactive solutions that use *probabilistically-checkable proofs* (PCPs) have since been developed such as Pepper [15] and Ginger [16].

Incentive-based solutions such as [2, 13] have studied contracts that the outsourcer may offer to the agents and through a combination of auditing, fines and rewards, honest computation is enforced. All of these verification techniques are, however, costly in terms of computation, memory, incentive rewards, etc., either to the prover or the verifier, or both. For example, the scheme in [12] requires partial re-execution of the tasks, and the verification in [3] incurs cost in the redundancy of the number of computing agents. Also, efficient protocols like Pepper still incurs a cost in the order of m^3 [15] on the principal, where m is the size of the problem. The cost of employing verifiable computing across these different schemes hence raises the important question of how to use them economically, especially when there is a flexibility in parameters that govern the overall cost to the outsourcer. Motivated by this, we abstract verification techniques as an auditing tool with a exogenous cost and provide incentive-based contracts that minimise the expected cost of the principal. Our contributions generalize the results in [2, 13] by (1) extending the feasibility of honesty enforcing schemes for *any* bound on the enforceable fines and *any* auditing capacity; (2) explicitly accounting for the cost of auditing and treating the auditing rate as one of the choice variables; and (3) providing optimal contract that minimize the aggregate cost of the principal as a combination of incentive payments and auditing costs. In short, our work explicitly extends both applicability and efficiency of incentive-based solutions based on a general abstraction of the verification method employed. For readers' interests, we also study in [10] the coalition among agents that may give them advantages in cheating the principal.

3 Problem Definition: General Setup

In this section, we describe the general setting of the problem and basic assumptions behind our model. A list of notations is provided in Table 1 for reference.

The outsourcer, which we refer to as the *principal*[1] has a deterministic *computation task* to be executed to obtain the output (result). Instead of executing the task itself, the principal hires a set of *agents*[2] to do this. The principal

[1] Also called the *boss* [2], *master* [6], *outsourcer* [4], *client* [8], *data owner* [13], etc.
[2] Also referred to as the *workers*, *servers*, *clouds*, or *contractors*.

aims to enforce *fully honest* computation of the task through setting a contract, involving rewards, auditing, and punishments (fines).

The principal and the agents are each selfish non-cooperative expected utility maximizers. Initially, we assume that everybody is risk-neutral, i.e., they have no strict preference between their expected utility and their utility of expected reward, and hence [9, ch.2.4], their utilities are linear function of the costs (with negative sign) and the rewards (with positive sign). Moreover, we assume that agents are "lazy but not malicious", that is, they do not have any interest in potentially reporting dishonest computations other than saving in their computation cost. Suppose the range and the probability distribution of the computation result is known. Generating a guessed output according to this distribution has zero computation cost and accuracy probability of q_0 (which can be negligibly small if the range of the output is large). For the sake of generality, as in [2], suppose each agent also has access to a private and independent *tricky algorithm* Alg that generates the correct output with probability q_1, where $q_0 < q_1 < 1$, at the cost of $c(q_1) \geq c(q_0) = 0$. The cost of honest computation is $c(1)$, which is strictly greater than $c(q_1)$. To enforce honesty of the agents, the principal *audits* the returned result with probability λ. We assume that auditing is perfect, i.e., if the output is indeed correct, the audit definitely confirms it (no "false positives"), and if the output is incorrect, the audit surely detects it (no "false negatives"). In the most basic contract, the principal decides on an auditing rate λ, sets a penalty (fine) f for detected erroneous answers and reward r otherwise. What make the problem non-trivial are the following observations:

1. *Costly detectability of cheating*: that auditing *all* of the results is either *infeasible* or *undesirable*. Regarding the infeasibility, suppose that in the long run the principal has a continuous demand (e.g. the Folding@Home project) of tasks awaiting computation, appearing at a rate ρ tasks per unit time. Also, suppose that each audit takes the principal ν machine cycles, and the computation capacity of the principal's machine is κ cycles per unit time. Then the maximum feasible rate of verification is $\frac{\kappa}{\nu\rho}$.[3] Moreover, auditing (e.g. through re-computation) may be costly as it will consume the computation power of the principal's machine and slow it down, or it will require obtaining additional hardware. The principal chooses the probability of auditing of a task $\lambda \in [0, \Lambda]$, where $0 < \Lambda \leq 1$ is associated with the computational capacity of the principal. The principal incurs the cost $\Gamma(\lambda)$ which is non-decreasing in λ. For simplicity of exposition, we assume a linear relation: $\Gamma(\lambda) = \gamma\lambda$ for a given $\gamma \geq 0$. An alternative to the occasional redoing of the whole computation by the principal can be using a third-party cloud that

[3] Note that even when the principal is verifying at full capacity, it should not pick the next immediate task to verify after finishing the previous one, since it may create a "learnable" pattern of audited tasks, which the agent can use to only be honest when computing them. This however can be avoided if the principal picks uniformly randomly tasks at the rate of $\frac{\kappa}{\nu\rho}$ and store them in a queue. However, the practical buffer has a storage limit. Consequently, the maximum feasible auditing rate with no essential pattern is strictly less than the full capacity rate $\frac{\kappa}{\nu\rho}$.

is highly reliable but costly (with per access cost of γ). For this scenario, the maximum auditing rate Λ is one, i.e., all of the tasks could be audited, albeit at an excessive cost.

2. *Limited enforceability of the fines*: The problem of verifiable computing could become trivial if there is no bound on the fine that can be practically levied on a wrongdoer: as long as there is even a tiniest probability of detection, then the principal can make the expected utility of the smallest likelihood of cheating become negative by setting the fine for erroneous results large enough. The issue with this argument is that such a fine may be extremely large and hence, become an *incredible threat*, in that, if the cheating of an agent is indeed caught, the fine is practically or legally non-collectable. Thus, existence (feasibility) results of honesty enforcement that rely on choosing a "large enough" fine are rather straightforward and uninteresting. In particular, such approaches leave unanswered the question of whether honest computation is still attainable for a bounded enforceable fine below their prescriptive threshold. Moreover, such results do not provide a good metric of comparison between alternative incentive schemes, or across different choices of parameters for a particular scheme. We will explicitly introduce $F \geq 0$ in our model to represent the maximum enforceable fine and obtain the optimal contracts subject to $f \leq F$. This can be the "security deposit", prepaid by the agent to the principal, that is collectible upon a provable detection of an erroneous result. A special case of interest is $F = 0$, i.e., when the only means of punishment is refusal to pay the reward.
3. *Limited budget*: As with the maximum enforceable fine to make it a credible threat, the maximum instantaneous "budget" of the principal leads to a bound on the reward to make it a credible promise. Let the maximum instantaneous payable reward by the principal be R. Thus, we require: $r \leq R$.

4 Contracts for Single Agent

In this section, we consider the case where the contract is designed for and proposed to only one computing agent. We provide the optimal contract for the basic model in subsection 4.1. In subsection 4.2, we investigate what happens if the risk-neutrality assumption of the agents is relaxed. Next in subsection 4.3, we comment on moderating against using tricky algorithms and clever guesses. Subsequently, in subsection 4.4, we discuss the optimal choice of the principal in the light of the optimal contracts theretofore developed. We close the case of single-agent in subsection 4.5 by generalising our results to contracts in which the principal is allowed to reward unaudited and verified tasks potentially differently. In Section 5, we will investigate the multi-agent case.

The action of the agent, given the parameters of the contract set by the principal, is first whether to accept it, and if so, which (probabilistic) algorithm to choose for computation of the assigned task. Since a naive random guess is correct with probability q_0, we assume that the agent's algorithm is correct with probability $q \in [q_0, 1]$. Let u_A denote the expected utility of the agent

Table 1: List of main notations

parameter	definition
λ	probability of auditing an outsourced computation by the principal
Λ	the physical upper-bound on λ
γ	cost of auditing (incurred by the principal)
q	probability of a correct computation by the agent
q_0	the correctness probability of a random guess from the output space
$c(q)$	the expected cost of computation to an agent for the correctness level of q
$c(1), c$	cost of an honest computation to an agent
f	fine collected from agent upon detection of an erroneous computation
F	the maximum enforceable fine
r	reward to the agent for an unaudited or audited and correct computation
R	the maximum feasible reward
z	the reserve utility (a.k.a., fallback utility or aspiration) of the agent
H	auxiliary coefficient defined as $c(1) + z$ (§4)
K	auxiliary coefficient defined as $(c(1) - c(q_1))/(1 - q_1)$ (§4)
$\mathcal{C}$	the expected cost of the contract to the principal
α	probability of using two agents for the same computation (§5.1)
F_0	auxiliary coefficient defined as $c/\Lambda - c$ (Proposition 5, §5.1)
F_1	auxiliary coefficient defined as $c[c-\gamma]^+/[2\gamma - c]^+$ (Proposition 5, §5.1)

after accepting the contract. With correctness probability of q, the agent is caught (and fined) with probability $(1-q)\lambda$. Hence, u_A is composed of expected reward $[1-(1-q)\lambda]r$, minus the expected cost composed of the cost $c(q)$ of the agent's algorithm and the expected fines $(1-q)\lambda f$. Hence: $u_A(q) = [1-(1-q)\lambda]r - c(q) - (1-q)\lambda f$. The agent may be able to achieve the same level of correctness, i.e., q, with different randomizations between the tricky algorithm Alg, the honest algorithm and random (naive) guessing. It is straightforward to make the following observation: For any q, the best $c(q)$ is achieved as follows:

a) If $[c(1) - c(q_1)]/(1-q_1) > c(1)/(1-q_0)$, then: $c(q) = \begin{cases} L_{q_0,q_1}(q) & q_0 \le q \le q_1 \\ L_{q_1,1}(q) & q_1 \le q \le 1 \end{cases}$;

b) If $[c(1) - c(q_1)]/(1-q_1) < c(1)/(1-q_0)$, then: $c(q) = L_{q_0,1}(q)$, where in both cases, $L_{x,y}(z) := c(x) + \dfrac{c(y)-c(x)}{y-x}(z-x)$, i.e., the linear combination of the costs of the corresponding two end points.

Note that in case-(b), the risk-neutral agent would never use Alg, since the cost of using it can be undercut (in expected value) by randomizing between honest computation and random guessing. Hence, we only consider case-(a) for now and revisit case-(b) in §4.3.

4.1 Optimum Contract for a Single Agent

The principal chooses the contract by setting the rate of auditing and reward and punishment values, in order to maximize its own utility and ensure *fully honest*

computation. Hence, the reward and punishments, r and f, should be chosen such that honest computation is the optimal course of action for the agent, if the contract is accepted. This means ensuring: $1 = \arg\max u_A(q)$. Following the Principal-Agent modelling in game theory (e.g. [9, ch.7] or [14, ch.6]), we will refer to this as the *incentive compatibility* constraint. For case (a), this becomes:

$$u_A(1) = r - c(1) \geq u_A(q_1) = [1 - (1 - q_1)\lambda]r - c(q_1) - (1 - q_1)\lambda f \qquad (1)$$

The agent accepts the contract if its expected utility is larger than its *reserve utility*, $z \geq 0$.[4] Given incentive compatibility, this *participation constraint* is hence:[5]

$$u_A(1) = r - c(1) \geq z. \qquad (2)$$

The principal wants to get away with the least reward and auditing rate. Therefore, the *optimal* contract for the single agent reduces to solution of the following optimization:

$$\min_{r,f,\lambda} \mathcal{C} := r + \gamma\lambda \qquad (3a)$$

$$s.t. \quad r \leq R,\ 0 \leq f \leq F,\ 0 \leq \lambda \leq \Lambda, \qquad (3b)$$

$$r \geq H,\ r\lambda + f\lambda \geq K \qquad (3c)$$

where (3c) is derived from (1) and (2) in which we have used the auxiliary coefficients $H := c(1) + z$ and $K := [c(1) - c(q_1)]/(1 - q_1)$ for brevity. Then:

Proposition 1. *With the parameters given in Table 1, the contract that enforces honest computation and is accepted by the agent, and minimizes the cost of the principal is by setting $f^* = F$ and choosing λ^*, r^* as given by the following:*[6]

$$\gamma \leq \frac{K}{\Lambda^2}: \begin{cases} [\frac{K}{\Lambda} - H]^+ \leq F: & \lambda^* = \frac{K}{H+F},\ r^* = H,\ \mathcal{C}^* = H + \frac{\gamma K}{H+F} \\ [\frac{K}{\Lambda} - R]^+ \leq F < [\frac{K}{\Lambda} - H]^+: & \lambda^* = \Lambda,\ r^* = \frac{K}{\Lambda} - F,\ \mathcal{C}^* = \frac{K}{\Lambda} + \gamma\Lambda - F \end{cases}$$

$$\gamma > \frac{K}{\Lambda^2}: \begin{cases} [\sqrt{K\gamma} - H]^+ \leq F: & \lambda^* = \frac{K}{H+F},\ r^* = H,\ \mathcal{C}^* = H + \frac{\gamma K}{H+F} \\ [\sqrt{K\gamma} - R]^+ \leq F < [\sqrt{K\gamma} - H]^+: & \lambda^* = \sqrt{\frac{K}{\gamma}}, r^* = \sqrt{K\gamma} - F, \mathcal{C}^* = 2\sqrt{K\gamma} - F \\ [\frac{K}{\Lambda} - R]^+ \leq F < [\sqrt{K\gamma} - R]^+: & \lambda^* = \frac{K}{R+F},\ r^* = R,\ \mathcal{C}^* = R + \frac{\gamma K}{R+F} \end{cases}$$

For $F < [\frac{K}{\Lambda} - R]^+$, the optimization is infeasible, i.e., there is no honesty-enforcing contract that is also accepted by the agent.

[4] The reserve utility (also referred to as the *fall-back utility* or *aspiration wage*) is the minimum utility that the agent aspires to attain or can obtain from other offers. Naturally, $z \geq 0$. Note that an implicit assumption here is that the agent is replaceable by any other agent with the same fall-back utility, i.e., there are many agents available with the same reserve utility. Without this assumption, the agent has negotiation power by refusing the contract knowing that it cannot be replaced. Alternatively, z can be thought as to (exogenously) capture the negotiation power of the agents. This is an assumption we make throughout the paper.

[5] Participation constraint is sometimes also called Individual Rationality constraint.

[6] The notation $x^+ := \max\{0, x\}$.

Discussion. The first observation is that the optimal contract should fully utilize the maximum enforceable fine and punish at no less than F. For large values of enforceable fines, we note that r^* is at H, the minimum value to ensure participation, and $\lim_{F\to\infty} \lambda^* = 0$, which yields $\lim_{F\to\infty} \mathcal{C}^* = H$. These are compatible with intuition as a huge fine implies that honesty can be enforced with minimum compensation and minuscule rate of inspection. When auditing is cheap ($\gamma \leq K/\Lambda^2$), increasing the auditing rate is the better option to compensate for lower values of F to maintain incentive compatibility (honest computation). This is unless the auditing rate is at its maximum Λ, in which case, reward must increase above H to maintain incentive compatibility and compensate for the low value of F. Note that in this case, the participation constraint is not active and is satisfied with a slack, while the incentive compatibility constraint is satisfied tightly. For yet lower values of enforceable fine F, even maximum reward $r = R$ and auditing rate $\lambda = \Lambda$ might not impose a strong enough threat against cheating, hence the infeasibility region. When auditing is expensive ($\gamma > K/\Lambda^2$), in order to retain incentive compatibility in the situation of very low fine F, the principal should increase reward, and only consider more frequent auditing if the reward budget R has been reached. Fig. 1 depicts the optimal parameters of the contract versus the maximum enforceable fine for the latter case ($\gamma > K/\Lambda^2$).

Note that the infeasible region does not necessarily exist. Specifically, when the principal's instantaneous budget R is larger than K/Λ, then there is always a feasible contract. Then even for $F = 0$, i.e., no enforceable fine, a contract that enforces honest computing is feasible, albeit by using high values of reward and/or auditing rate. In such cases, the principal "punishes" audited erroneous computations only through not rewarding the agent. However, it is clear that honesty cannot be enforced with zero auditing rate, and hence the case of $\Lambda = 0$ trivially leads to infeasibility. Moreover, to satisfy the participation constraint at all, R has to be at least as large as H. Hence, for $R < H$, likewise, there exists no feasible contract for any F. We also show that except for the special case of $\gamma = 0$, the optimal contract has the feature that it is *unique*. Figures 2a and 2b depict

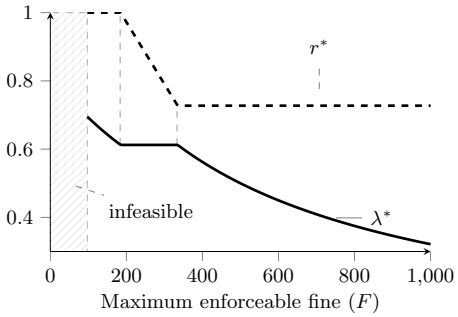

Fig. 1: Change of contract parameters r^*, λ^* w.r.t. the maximum enforceable fine F (Prop. 1, case of $\gamma > \frac{K}{\Lambda^2}$), where $K = 450$, $\gamma = 1200$, $\Lambda = 0.7$, and $c = 400$.

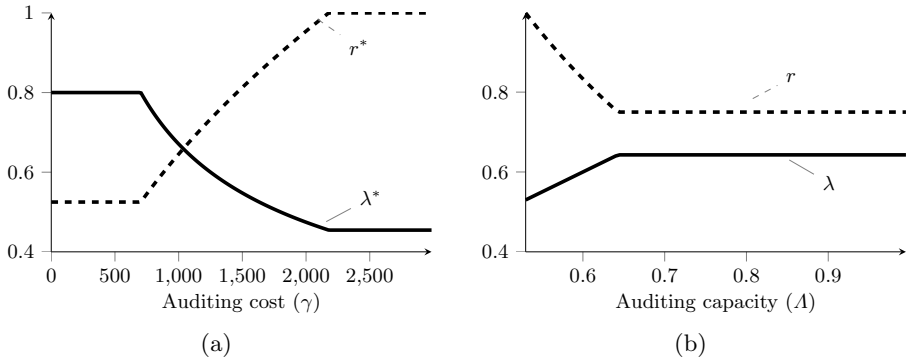

Fig. 2: Optimal contract parameters w.r.t (a) the auditing cost γ, with $K = 450$, $\Lambda = 0.8$, $c = 400$, and (b) auditing capacity Λ, with $K = 450$, $\gamma = 450$, $c = 450$.

the change in the structure of the optimal contract versus varying auditing cost γ and the maximum auditing capacity, respectively. From Fig. 2a, we can see that for larger values of γ, the optimal contract utilizes lower values of inspection rate λ^* while using higher values of reward r to enforce honest computation. This transition progress culminates when the payment reaches its threshold R, after which the contract remains unchanged. In contrast, Fig. 2b shows how increasing the maximum auditing capacity affects the optimal contract in the opposite trend: as the principal is more capable of auditing, it should consider more frequent auditing and lessen the reward for honest computation. The payment, however, can never be lowered below H to maintain participation.

4.2 A Risk-Averse Agent

So far, we modelled the agent as risk-neutral, i.e., one that is indifferent between its expected utility and utility of expectation, leading to a linear utility function. However, empirically, individuals tend to show risk-aversion regarding decisions that affect their income. By definition, (strict) risk aversion is (strict) preference of expected utility over utility of expectation. Following *Jensen*'s inequality, this is equivalent to assuming a (strictly) concave utility function (ref. e.g. [9, ch.2.4]). We have the following simple but re-assuring result:

Proposition 2. *The optimal contract given in Proposition 1 developed for a risk-neutral agent stays feasible for any risk-averse agent as well.*

Note that even though the feasibility of our contract is guaranteed, its optimality might no longer hold. This is because a lower value of fine and/or rewards could potentially maintain incentive compatibility, as intuitively, cheating with a chance of getting caught can be seen as a lottery. However, because the level of risk-averseness of an agent is unknown, we argue that it is best practice to design the optimal contract for the worst case with respect to risk, i.e., risk neutrality.

Specially, if a contract is designed assuming a particular degree of risk-aversion of the agent but the agent turns out to be less risk-averse than assumed, then the incentive-compatibility for honest computation may be violated, failing the principal's intolerance of erroneous computations. Accordingly, for the rest of the paper, we will retain risk-neutrality for agents.

4.3 Mitigating clever guesses

An inherent problem of outsourced computation is that a (not always) correctly guessed output is indistinguishable from an honestly computed one. For instance, consider the question of whether a large natural number is a prime: the deterministic guess of "no" is most likely correct. Also, since the principal might not know the exact cost and success probability of potential guessing algorithms, it is hard to design a contract that enforces honesty. Therefore, the principal may prefer to avoid identifying the parameters of guessing algorithms altogether.

One way to mitigate the possibility of "clever" guesses is to enlarge the output range by requiring the agent to return not just the final computation output, but also snapshots of intermediate steps of the computing process [2]. This will reduce the correctness probability of a naive guess down to $q_0 = \mathsf{negl}$. Moreover, requiring snapshots of the intermediate steps makes guessing of the correct output more costly. Let $c(q_1)$ be the cost of a tricky algorithm that tries to produce the expanded output with the intermediate steps of the honest computation, where it succeeds with probability q_1. We make the assumption that now $c(q_1) > q_1 c(1)$, so that any guessing algorithm with cost $c(q_1)$ can be replaced with a randomization between naive guess (with weight $1 - q_1$) and honest computation (with weight q_1). Thus, for incentive compatibility, we only need to make sure that the agent's utility from honest computation is better than a naive guess that succeeds with negligible probability $q_0 = \mathsf{negl}$. To avoid distraction in our analysis, we assume $q_0 = 0$, as the results can easily be realized for $q_0 = \mathsf{negl}$. Our simplified constraints for the contract become:

$$\text{participation}: \ r \geq c(1) + z, \ \text{incentive compatibility}: \ r \geq \frac{1}{\lambda} c(1) - f. \qquad (4)$$

Comparing to the constraints in (3c), this translates to changing K to $c(1)$. This in turn implies that the new incentive compatibility constraint requires a strictly lower fine value. Intuitively, as guessing becomes more difficult, cheating becomes less attractive and hence can be deterred with a smaller fine. Hereafter, we assume that the principal is employing this technique and use the above incentive compatibility constraint. Moreover, for simplicity of exposition, we assume that the reserve utility z is zero, and hence H becomes $c(1)$, which we will abbreviate as c.

4.4 Optimal Choice for the Principal

So far we have considered auditing as a blackbox and only included its cost and capacity into the model. However, when auditing is via redoing the computation

(at the cost of γ) it might be optimal for the principal to not offer any contract at all. Indeed, when $\Lambda = 1$, the principal can potentially audit all computations by redoing them. Specifically, if the optimal contract costs $\mathcal{C}^* \geq \gamma$, then it is optimal for the principal to do the computation itself, as that only costs $\gamma\Lambda = \gamma$. In case $\Lambda < 1$, the principal cannot *do* all the computations, and must outsource a portion of it. Interestingly, the following proposition establishes that the principal's optimal choice is either to not outsource at all, or fully outsource its computation.

Proposition 3. *Consider the case where auditing is through redoing the computation. Let x be the probability that the principal computes the tasks itself. Then, either $x^* = 0$ and the optimal contract is as per Proposition 1, or $x^* = \Lambda = 1$ and there should be no outsourcing.*

The proposition has this important corollary:

Corollary 1. *When $\Lambda < 1$, the optimal choice for the principal is to use the optimal contact given by Proposition 1. When $\Lambda = 1$, the optimal choice of the principal is to compare the expected cost achieved by the optimal contract in Proposition 1 (for the value of maximum enforceable fine at hand) against γ, and accordingly decide to outsource or independently compute all of the tasks.*

4.5 Optimal Contract for a Single Agent: Two-Level Reward

In our contracts so far, verified correct results and unaudited results are rewarded identically at r. Suppose, alternatively, that the principal rewards r_0 for accepted but not audited results and r_1 for corroborated correct answers, and as before, penalizes f for detected wrong computations. This way, the principal may hope to save significantly by, for example, not paying for unaudited computations. The new incentive compatibility and participation constraints are: $(1-\lambda)r_0+\lambda r_1-c \geq (1-\lambda)r_0 - \lambda f$ and $(1-\lambda)r_0 + \lambda r_1 - c \geq 0$, respectively. The optimization of (3) for a contract with two-level reward changes to:

$$\min_{r_0, r_1, f, \gamma} \mathcal{C} := r_1\lambda + r_0(1-\lambda) + \gamma\lambda$$
$$s.t. \quad r_0, r_1 \leq R,\ f \leq F,\ 0 \leq \lambda \leq \Lambda,\ r_1\lambda + r_0(1-\lambda) \geq c,\ r_1 \geq \frac{c}{\lambda} - f.$$

Proposition 4. *For $F \geq [c/\Lambda - R]^+$, the optimal single-agent contract for two-level rewarding is given as: $f^* = F$, $\lambda^* = c/(F+R)$, $r_1^* = R$, $r_0^* = Fc/(R-c+F)$, $\mathcal{C}^* = c\,(1 + (\gamma + c - R)/(F+R))$. For $F < [c/\Lambda - R]^+$, the contract is infeasible.*

Discussion of the two level reward contract. First, note that there is no improvement in terms of the infeasibility region compared with the single-level reward contract. However, the achieved cost is always better. This was to be expected as the single-level rewarding can be thought of as a special case of two-level. However, the behaviour of the optimal contract now does not depend on the value

of the auditing cost γ. This is where the strength of the two-level rewarding lies: for high values of γ, the two-level contract increasingly outperforms the single reward-level contract.

Note that the optimal reward for audited and correct results r_1 is at the principal's maximum budget R irrespective of the value of F. The value of reward for unaudited results r_0 is always strictly less than c, i.e., the cost of honest computation (and hence strictly less than r_1 as well). The value of r_0, unlike r_1, depends on F: For higher values of maximum enforceable fine, in fact somewhat unexpectedly, the optimal contract chooses increasing values of reward r_0^*. Still intuitively, a larger threat allows less necessity for auditing, and thus the contract starts to behave as a "lottery", in which the low-chance "winner" receives $r_1^* = R$ and the "loser" $r_0 < c < R$.

5 Optimal Contracts for Multiple Agents

When there are more than one agent available, the set of possible contracts gets extended. Specifically, as e.g. [2] and [13] discuss, the principal has the option of submitting the same task to multiple agents and comparing the outcomes. We will refer to this option as the *redundancy* scheme. If the returned results do not match, it is clear that at least one agent is cheating. Furthermore, as [13] assumes, if the agents are non-colluding, and returning the intermediate steps along with the computation result is required, then the probability that the results produced by cheating will be the same will be negligible, which we again assume to be zero (for simplicity). Hence, the returned results are correct *if and only if* they are the same.

In the next subsection, we develop optimal contracts considering two agents. Subsequently, we establish the global optimality of two-agent contracts among any number of agents with respect to the notion of Nash Equilibrium.

5.1 Optimal Contracts for Two Agents

Consider the case that there are two agents available: agent 1 and 2. As in the single-agent case, consider a principal that has a computation task and a maximum auditing rate of Λ. Then, in general, a principal can use a hybrid scheme: it may choose to send the same job to both of the agents sometimes, and to one randomly selected agents the rest of the time. Sending the same task to two agents provides a definite verification, however, at the cost of paying twice the reward, since both agents must be rewarded for honest computation. Hence, an optimal choice of redundancy scheme is not immediately clear, even less so if this schemes is randomized with just choosing one agent and doing independent audits. In this section, we investigate optimal contracts among all hybrid schemes.

Besides lack of collusion, we assume the agents do not communicate either. Therefore, on the event that any of the agents receives a task, it has no information about the busy/idle state of the other agent. The action of each agent is

selection between honest computation, which we represent by $\mathscr{H}$, and cheating, which we denote by $\mathscr{C}$. Since the agents have no information about the state of the other agent, the set of their (pure) strategies and actions are the same.

The expected utility of each agent depends in part on the action of itself and of the other agent. Let $u_A(a_1, a_2)$ represent the utility of agent 1 when it chooses action a_1 and agent 2 chooses a_2, where $a_1, a_2 \in \{\mathscr{H}, \mathscr{C}\}$. The principal wants to enforce honest computation with probability one. If $u_A(\mathscr{H}, \mathscr{H}) \geq u_A(\mathscr{C}, \mathscr{H})$, then given that agent 2 is going to be computing honestly, agent 1 will prefer to do the same too, and due to symmetry, likewise for agent 2. In the game theoretic lingo, this means that $(\mathscr{H}, \mathscr{H})$ is a (Nash) equilibrium. If, further, $u_A(\mathscr{H}, \mathscr{C}) \geq u_A(\mathscr{C}, \mathscr{C})$, then $(\mathscr{H}, \mathscr{H})$ will be the dominant (Nash) equilibrium, i.e., honest computation is the preferred action irrespective of the action of the other agent.

The principal utilizes the redundancy scheme with probability α or employs only one of the agents (selected equally likely)[7] with probability $1 - \alpha$. If the principal chooses only one agent, then it audits it with probability ρ. Since auditing only occurs when a single agent receives the task, the likelihood λ that the task will ever be audited is $\rho(1 - \alpha)$. As in the single-agent single-reward scenario, if only one agent is selected, the agent is rewarded r if there is no indication of wrongdoing, and is punished f if audited and caught wrong. When the redundancy scheme is selected and the returned results are equal, both agents are rewarded r. Otherwise, both are fined at f. With the model so described, the expected utilities of an agent are computed as follows:[8]

$$u_A(\mathscr{H}, \mathscr{H}) = r - c, \qquad u_A(\mathscr{C}, \mathscr{H}) = (1 - \alpha - \lambda)r/2 - (\alpha + \lambda/2)f.$$

Hence, the condition $u_A(\mathscr{H}, \mathscr{H}) \geq u_A(\mathscr{C}, \mathscr{H})$ becomes: $r \geq (1 + \alpha)c/(\lambda + 2\alpha) - f$. Subject to making $(\mathscr{H}, \mathscr{H})$ an equilibrium, the contract is accepted if the expected utility of it to the agents is above their reserve utility, which we assume here too to be zero for simplicity: $r - c \geq 0$. Then the expected cost of the contract to the principal is:

$$\mathcal{C} = 2r\alpha + \gamma\lambda + r(1 - \alpha) = (1 + \alpha)r + \gamma\lambda.$$

The principal chooses λ, α, f, r such that honest computation is enforced, the contract is accepted, and the expected cost of the principal is minimized. λ and α must satisfy the structural condition $0 \leq \alpha \leq 1$, $0 \leq \lambda \leq \Lambda$ and $\alpha + \lambda \leq 1$. The instantaneous budget of the principal imposes $r \leq R$ if $\alpha = 0$, and $2r \leq R$ if $\alpha > 0$. We assume $R \geq 2c$, since otherwise, the principal can never employ both of the agents without violating its instantaneous budget constraint, and hence, the problem reduces to the single agent problem. Then, the budget constraint

[7] We will formally show through the proof of proposition 6 that equal randomization is the best option. Intuitively, this removes any information that the agents may infer upon receiving a task.

[8] Since the only information state to an agent is whether it receives the job, the ex-ante and ex-post analysis, i.e., before and after reception of the task, become equivalent. We present the ex-ante view for simplicity.

simplifies to $r \leq R/2$. Therefore, the optimal contracts for two agents that make $(\mathscr{H}, \mathscr{H})$ an equilibrium are solutions of the optimization problem of:

$$\min_{r,f,\alpha,\lambda} r(1+\alpha) + \gamma\lambda \text{ subject to:}$$

$$r \leq R/2,\ f \leq F,\ 0 \leq \lambda \leq \Lambda,\ \lambda \leq 1-\alpha,\ \alpha \geq 0,\ r \geq c,\ r \geq \frac{c(1+\alpha)}{\lambda + 2\alpha} - f.$$

Note that the above optimisation only guarantees that $(\mathscr{H}, \mathscr{H})$ is a Nash equilibrium. Other strategy profiles might become equilibria, for example $(\mathscr{C}, \mathscr{C})$. However we notice that because agents are only rewarded when they are both honest, $(\mathscr{H}, \mathscr{H})$ is thus the most attractive equilibrium to agents both individually and socially. We therefore only care to ensure that $(\mathscr{H}, \mathscr{H})$ is an equilibrium. The optimal contracts are as follows:

Proposition 5. *Let $F_0 = c/\Lambda - c$ and $F_1 = c[c-\gamma]^+/[2\gamma - c]^+$,[9] the optimal one-level reward two-agent contract that makes $(\mathscr{H}, \mathscr{H})$ a Nash equilibrium is:*

$$\begin{cases} F_1 \leq F: & f^* = F,\ \alpha^* = \frac{c}{2F+c},\ \lambda^* = 0, r^* = c,\ \mathcal{C}^* = c(1+\frac{c}{2F+c}) \\ F_0 \leq F < F_1: & f^* = F,\ \alpha^* = 0,\ \lambda^* = \frac{c}{c+F},\ r^* = c,\ \mathcal{C}^* = c(1+\frac{\gamma}{F+c}) \\ F < \min(F_0, F_1): & f^* = F,\ \alpha^* = \frac{c - \Lambda(c+F)}{c+2F},\ \lambda^* = \Lambda,\ r^* = c,\ \mathcal{C}^* = \frac{c(c+F)(2-\Lambda)}{c+2F} + \gamma\Lambda \end{cases}$$

For $\Lambda = 1$, $(\mathscr{H}, \mathscr{H})$ is moreover the dominant Nash equilibrium.

Corollary 2. *If auditing is more expensive than the cost of honest computation ($\gamma \geq c$), the optimal contract only uses the redundancy scheme. When $\gamma \leq c/2$, either there is no redundancy scheme ($\alpha = 0$) or the whole auditing capacity is used ($\lambda^* = \Lambda$).*

The first part of the corollary is quite intuitive: when $\gamma > c$, any instance of outsourcing to a single agent and performing independent auditing can be replaced by the redundancy scheme (job duplication) and strictly lower the cost by $\gamma - c$.

Further Discussion. First, note that in our optimal two-agent contract, as long as $R \geq 2c$, there is no infeasible region: there is always a contract that makes $(\mathscr{H}, \mathscr{H})$ an equilibrium. Moreover, the payment to any of the agents is never more than the cost of honest computation. Fig. 3a provides a pictorial representation of the proposition where $c/2 < \gamma < c$ and $\Lambda = 0.5$. When the enforceable fine is large, the redundancy scheme is preferable. This is despite the fact that the redundancy scheme is more expensive than auditing: it costs an extra c as opposed to $\gamma < c$. In other words, for high values of fine, the redundancy scheme is a more effective threat against cheating than independent auditing. When F is less than F_1, the independent auditing becomes the preferred method. For lower values of F, when the auditing capacity is all used up, the redundancy scheme is added to compensate for the low value of fine to maintain incentive compatibility. Fig. 3b depicts the effect of auditing capacity, Λ, on the optimal contract

[9] We adopt the convention that $x/0 = +\infty$ for $x > 0$.

where $c/2 < \gamma < c$. When $\Lambda = 0$, redundancy scheme is the only means to enforce honest computation. If furthermore no fine can be enforced ($F = 0$), then $\alpha = 1$: the job should be always duplicated. As Λ increases, there is a gradual transition from using redundancy scheme to independent auditing ($F < F_1$).

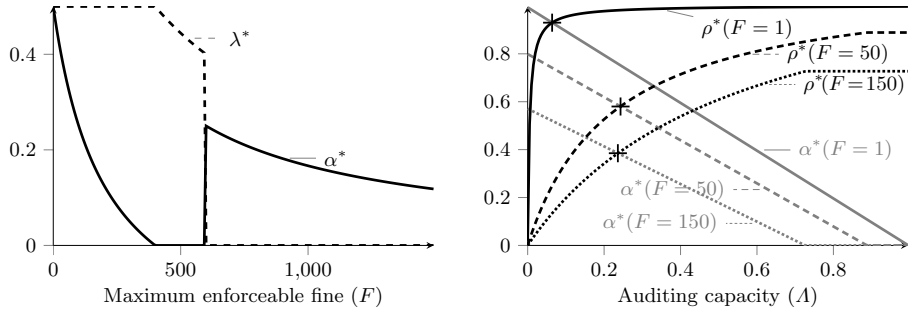

Fig. 3: Optimal contract (where $c = 400, \gamma = 250$) w.r.t. (a) max. enforceable fine F ($\Lambda = 0.5$); and (b) auditing capacity Λ ($F_1 = 600$). Recall $\rho = \frac{\lambda}{1-\alpha}$ is the conditional probability of auditing given the job is assigned to a single agent.

5.2 Global Optimality of Two-Agent Contracts

In developing the optimal contracts for two-agent case, we made a few critical assumptions: (a) the independent auditing is perfect; (b) the agents are non-colluding and non-communicating; (c) the range of intermediate steps is large enough that the probability of any two guessed results to be same, or the guessed result to be the correct result, is negligible; and (d) the agents are lazy but not malicious. It turns out that these assumptions are sufficient to warrant global optimality of two-agent contracts among contracts that engage any number of agents in the following notion:

Proposition 6. *The contract that hires at most two agents and chooses its terms according to proposition 5, is globally optimal, that is, it achieves the least cost to the principal among all contracts that employ any number of agents and aim to make honest computation a Nash Equilibrium.*

The above proposition shows that our contract for two agents is not just a special case solution of multiple agents, but it is indeed the solution involving any number of agents. In other words, given the stipulated assumptions, there is no advantage ever in hiring more than two agents. Incidentally, we also show that the best contracts makes the probability of any of the agents to be hired equal. This makes intuitive sense, as unequal probability of task assignment creates some "information" which the agents can potentially exploit to their benefit, and to the detriment of the principal.

6 Contract Implementation

For completeness of the solutions, in this section we discuss notable technical concerns on the implementation of our contracts.

6.1 Intermediate steps and hash functions

As we discussed in Section 4.3, the use of intermediate steps as part of the output would prevent trivial/clever guessing. However, the data representing intermediate steps could be large and thus cumbersome for transmission. [2] proposes the use of cryptographic hash as a sufficient representation of intermediate steps: Instead of sending a large amount of data detailing these steps, the agent can only send the cryptographic hash of such data. On receiving the agent's hash h_A, the principal repeats the computation, and computes its own hash h_P from the intermediate steps, then verifies that $h_A = h_P$.

Informally, the use of hash function is considered secure if it is unlikely that the agent can come up with the correct hash without knowing the correct intermediate steps. The authors in [2] require such hash function to be a "random oracle", i.e., a function mapping in which each output is chosen uniformly randomly regardless of the input. While this is a sufficient condition, the notion of random oracle is rather impractical, and also an overkill. Indeed, we argue that for this purpose of hash checking, it is necessary and sufficient that the hash function is "collision resistant", that is, it should be difficult to find two different messages with the same hash.

Lastly, note that the process of hashing the intermediate steps may itself carry a considerable cost. For instance, if the computation task is to hash a large string, then the cost of hashing the intermediate steps (if the same hash function is used) would be at least as much as computation cost. Therefore, either the cost of hasing intermediate steps must be negligible compared to that of the original computation task, or it must enter the contract model.

6.2 Enforcing contract policies

With regards to legal enforcement of the contract, it is necessary that behaviours of contract participants are observable and verifiable. Actions such as "assigning a job" or "paying a reward" are of this type. However, probabilistic behaviours, e.g., "employing two agents with probability α", are usually unverifiable. Our contracts unfortunately rely on these probabilistic actions of the principal as explicitly stated in the terms and policies for auditing, task duplication and/or rewarding (the latter in two-level reward contracts of §4.5). It is critical to ensure (by both ends) that the principal in reality sticks to such actions, for two reasons. Firstly, the principal must establish to the agents its compliance to the contract so as to make the threats credible. Secondly, the agent needs an assurance that the principal cannot deviate from the contract and thus take away some of its benefits (in two-level rewarding). Without an appropriate security measure, this is usually not possible, e.g., the fact that the principal does not audit tells

little about whether its auditing probability is indeed $\lambda = 0.3$ or $\lambda = 0.6$. This important implementation issue has not been discussed in previous works.

Usually this could be achieved cryptographically using multiparty computation (MPC) [11], in which a sampling function on the principal's behaviour is accurately and securely computed among the contract participants. However, MPC assumes pairwise secure communication among participants, which in this case implies a need for direct communication between the agents. This poses a potential threat to our model: if agents can freely communicate, they may as well collude and give identically incorrect result, thus fooling the principal. Therefore we seek a mechanism that requires no agent-to-agent communication. In Fig. 4, we propose a communication protocol between the principal and two agents that resolves this problem. Particularly, our security objective is to make sure that the principal gains negligible benefit by deviating from its prescribed behaviour as stated in the contract. To fulfil this objective, we rely on the fact that the contract can be legally enforced by an authority (e.g., a court), and thus punishment on the principal's cheating is guaranteed if there is enough evidence for the accusation. What remains is to ensure that each agent alone can prove the principal's deviation (from the contract) whenever the principal benefits from doing so.

Protocol ContractProtocol

Requirement A (non-interactive) commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ and a trusted third party TTP. An optimal contract $\langle r, f, \alpha, \lambda \rangle$, a computation task J, and a security parameter $k > 0$.

Preparation The contract is signed by all parties with an additional term: if the principal is caught deviating from the below protocol, it must pay the worst possible cost [10]. TTP generates CK from $\mathsf{Setup}(k)$ and gives it to the principal (P), agent 1 (A_1) and agent 2 (A_2).

Protocol
1. A_1: generates $N_1 \leftarrow_{\$} \{0,1\}^k$, computes $(c_1^*, d_1^*) = \mathsf{Commit}_{\mathsf{CK}}(N_1)$, then sends c_1^* to P.
2. A_2: generates $N_2 \leftarrow_{\$} \{0,1\}^k$, computes $(c_2^*, d_2^*) = \mathsf{Commit}_{\mathsf{CK}}(N_2)$, then sends c_2^* to P.
3. P: sends (c_1^*, c_2^*) to both A_1 and A_2.
4. A_1: sends d_1^* to P.
5. A_2: sends d_2^* to P.
6. P: opens $N_1 \leftarrow \mathsf{Open}_{\mathsf{CK}}(c_1^*, d_1^*)$, $N_2 \leftarrow \mathsf{Open}_{\mathsf{CK}}(c_2^*, d_2^*)$, compute $\omega \leftarrow \mathsf{Gen}_{\Delta(\Omega)}(N_1 \oplus N_2)$, and follows plan ω.
7. P: sends (d_1^*, d_2^*) to both A_1 and A_2.
8. A_1, A_2: open $N_1 \leftarrow \mathsf{Open}_{\mathsf{CK}}(c_1^*, d_1^*)$, $N_2 \leftarrow \mathsf{Open}_{\mathsf{CK}}(c_2^*, d_2^*)$, compute $\omega \leftarrow \mathsf{Gen}_{\Delta(\Omega)}(N_1 \oplus N_2)$, and check if the principal follows ω.

Fig. 4: Communication protocol for the contract

In order to provably design such mechanism, we define the principal's action as a *plan*, which essentially captures the its deterministic choices for all possible decision-making situations which might arise while executing the contract. An example of such plan could be: give the task to both agents; if the result coming back is the same, then reward both. Another example is: give the task to agent 1, then audit on return. For convenience we denote the set of all possible plans as Ω, which also contains an element $\perp$ representing an invalid plan. The principal P is supposed to pick a plan $\omega \in \Omega$ according to a contract-specific probability distribution $\Delta(\Omega)$, but the agents do not know if P actually follows this distribution, or a different one to its eventual benefit. As a result, we decide to let such a plan be picked by the agents instead of the principal. The protocol for "picking plan" should satisfy the following properties:

- **Correctness**: Honest execution of the protocol must ensure that the plan is picked according to $\Delta(\Omega)$.
- **Hiding**: Before the contract is executed, the agents must know nothing about the plan they have picked for the principal.
- **Revealing**: After the contract is executed, there must be a secure way for the previously picked plan to be revealed to the agents.
- **No cheating**: Suppose that the agents execute the protocol honestly, then the principal receives no better benefit than being a honest principal.

We are now ready to construct our contract implementation protocol. For each probability distribution $\Delta(\Omega)$ assume that there exists a PPT *contract-generation* algorithm $\mathsf{Gen}_{\Delta(\Omega)}(\cdot)$ which efficiently samples $\Delta(\Omega)$, that is, there exists a negligible function ϵ_{G} and $k_1 > 0$ such that for all $k \geq k_1$:

$$\sup_{o \in \Omega} \left| \Pr\left[r \leftarrow_{\$} \{0,1\}^k; \omega \leftarrow \mathsf{Gen}_{\Delta(\Omega)}(r)\right] - \Pr\left[\omega \leftarrow \Delta(\Omega)\right] \right| \leq \epsilon_{\mathsf{G}}(k). \quad (7)$$

Whilst the protocol construction can be seen in Fig. 4, its security is described in Proposition 7. In words, the protocol involves the agents independently generate at uniformly random nonces N_1 and N_2, respectively. The agents then exchange these values using a commitment scheme via the principal P. The use of commitment ensures that even if the principal is able to modify the messages, it must not be able to convince each agent A_i of a nonce from the other which is dependent of N_i. This ensures that when A_i perform $N_1 \oplus N_2$ it would get a uniformly random value. Given the above property of $\mathsf{Gen}_{\Delta(\Omega)}$ the agent would receive a plan ω in the same distribution implied by the contract, thus avoid meaningful cheating by the principal.

Proposition 7 (informal). *Suppose all participants in* ContractProtocol *are PPT algorithms. Suppose that (*Setup, Commit, Open*) is a secure non-malleable commitment scheme, and that contract terms can be legally enforced and that both agents are honest, then* ContractProtocol *satisfies the following properties: correctness, hiding, revealing, and no cheating.*

[10] Here the worst possible cost (including what has been spent) is $\max(2r, r + \gamma)$, and it could either be distributed to the agents, or paid to the court as fine.

7 Conclusion

In this paper, we provide an incentive analysis of outsourced computation with non-malicious but selfish utility-maximising agents. We design contracts that minimise the expected cost of the outsourcer whilst ensuring participation and honesty of computing agents. We incorporate important real-world restrictions, in that the outsourcer can only levy a restricted fine on dishonest agents and that auditing can be costly and/or limited. We allow partial outsourcing, direct auditing and auditing through redundancy, i.e., employing multiple agents and comparing the results, and optimized the utility of the outsourcer among all hybrid possibilities.

We observe that outsourcing all or none of the tasks is optimal (and not partial outsourcing). We show that when the enforceable fine is restricted, achieving honest computation may still be feasible by appropriately increasing the reward above the sheer cost of honest computation. We demonstrate that when auditing is more expensive than the cost of honest computation, redundancy scheme is always the preferred method, and when the auditing cost is less than half of the cost of honest computation, independent auditing is preferable. When the cost of auditing is between half and the full cost of honest computation, the preferred method depends on the maximum enforceable fine: for large enforceable fines, redundancy scheme is preferred despite the fact that it is more expensive "per use" than independent auditing, since owing to its higher effectiveness, it can be used more sparingly. We establish the global optimality of contracts involving at most two agents among any arbitrary number of agents as far as implementing honesty as a Nash Equilibrium is aimed for. Finally, we present a light-weight cryptographic implementation of our contracts that provides mutual affirmation on proper execution of the agreed terms and conditions.

Acknowledgement

This research was partially sponsored by US Army Research laboratory and the UK Ministry of Defence under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Mikhail J Atallah, YounSun Cho, and Ashish Kundu. Efficient data authentication in an environment of untrusted third-party distributors. In *IEEE ICDE*, 2008.
2. Mira Belenkiy, Melissa Chase, C Chris Erway, John Jannotti, Alptekin Küpçü, and Anna Lysyanskaya. Incentivizing outsourced computation. In *NetEcon*. ACM, 2008.

3. Ran Canetti, Ben Riva, and Guy N Rothblum. Practical delegation of computation using multiple servers. In *ACM CCS*, 2011.
4. Bogdan Carbunar and Mahesh V Tripunitara. Payments for outsourced computations. *IEEE Transactions on Parallel and Distributed Systems*, 23(2), 2012.
5. Hong Chen, Xiaonan Ma, Windsor Hsu, Ninghui Li, and Qihua Wang. Access control friendly query verification for outsourced data publishing. In *ESORICS*, 2008.
6. Evgenia Christoforou, Antonio Fernández Anta, Chryssis Georgiou, Miguel A Mosteiro, and Angel Sánchez. Applying the dynamics of evolution to achieve reliability in master–worker computing. *Concurrency and Computation: Practice and Experience*, 2013.
7. Norton Rose Fullbright. Outsourcing in a brave new world: An international survey of current outsourcing practice and trends. Technical report, 2011.
8. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*. Springer, 2010.
9. Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2009.
10. MHR. Khouzani, Viet Pham, and Carlos Cid. Incentive engineering for outsourced computation in the face of collusion. In *Proceedings of WEIS 2014 – 13th Annual Workshop on the Economics of Information Security*, 2014.
11. Ueli Maurer. Secure multi-party computation made simple. In *Security in Communication Networks*, pages 14–28. Springer, 2003.
12. Fabian Monrose, Peter Wyckoff, and Aviel D Rubin. Distributed execution with remote audit. In *NDSS*, 1999.
13. Robert Nix and Murat Kantarcioglu. Contractual agreement design for enforcing honesty in cloud outsourcing. In *GameSec*. Springer, 2012.
14. E Rasmusen. Games and information: an introduction to game theory. 1994.
15. Srinath Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *NDSS*, 2012.
16. Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security*, 2012.
17. Cong Wang, Kui Ren, and Jia Wang. Secure and practical outsourcing of linear programming in cloud computing. In *INFOCOM, 2011*, 2011.
18. Ke Yi, Feifei Li, Graham Cormode, Marios Hadjieleftheriou, George Kollios, and Divesh Srivastava. Small synopses for group-by query verification on outsourced data streams. *ACM TODS*, 2009.