Institut Eurécom
Corporate Communications Department
2229, route des Crêtes
B.P. 193
06904 Sophia Antipolis
FRANCE

Research Report RR-07-189

A Game Theoretic Model of a Protocol for Data Possession Verification

8 February 2007

Nouha Oualha, Pietro Michiardi and Yves Roudier[1]

Tel: (+33) 4 93 00 81 00
Fax: (+33) 4 93 00 82 00
Email: {oualha, michiardi, roudier}@eurecom.fr

[1] Eurecom's research is partially supported by its industrial partners: BMW Group Research & Technology – BMW Group Company, Bouygues Telecom, Cisco Systems, France Telecom, Hitachi, SFR , Sharp, STMicroelectronics, Swisscom, Thales

A Game Theoretic Model of a Protocol for Data Possession Verification

Nouha Oualha, Pietro Michiardi and Yves Roudier

Abstract

This report discusses how to model a protocol for the verification of data possession that is intended to secure a cooperative peer-to-peer storage application. The verification protocol is a primitive for cooperation assessment, and indirectly motivates nodes to behave cooperatively within the application. This capability of the protocol to enforce cooperation between a data holder and a data owner is proved theoretically by modeling the verification protocol as a Bayesian game, and demonstrating that the solution of the game is an equilibrium where both parties, owner and holder, are cooperative.

A Game Theoretic Model of a Protocol for Data Possession Verification

Nouha Oualha, Pietro Michiardi and Yves Roudier

1. Introduction

Applications for peer-to-peer data storage are gaining a lot of interest in the research community especially in the context of mobile ad hoc networks. The capabilities of today's multimedia enabled mobile devices together with the need to ubiquitously (and durably, contrary to peer-to-peer file-sharing) access one's own data result in the need to manage an increasingly huge amount of sensitive and critical information. The tendency to network self-organization together with the larger scale of such systems make it necessary to implement data management services such as distributed data storage in a cooperative fashion. Cooperative storage applications are designed to share the storage resources of network nodes for the common good of everyone (examples are PAST [1], OceanStore [2], FreeHaven [3] and FARSITE [4]). However, data are exposed to new threats, since cooperation between nodes is not guaranteed. Non-cooperation results in a new form of denial of service called selfishness: regarding data storage, nodes may discard some data they promised to store for other nodes in order to optimize their resource usage. Cooperation also raises the more usual issue of maliciousness whereby nodes aim at disrupting the system, for instance destroying data replicas as an attack to the storage service. Because of the dynamics of such systems and the scattered peer-to-peer storage of data, checking that some information is still available somewhere is quite more complex and a longer-term operation than checking that a route has been established with another node in multi-hop MANETs for instance. All these problems contribute to the difficulty of properly determining the actual availability of some data, stored and replicated among a set of peers.

As summarized in Section 2, we propose a verification protocol for such peer-to-peer storage problems based on challenge-response messages between the data owner node and the data holder node. The contribution of this report is the validation of this security primitive with respect to the cooperation enforcement of the storage of some data on one node. This primitive makes it possible for the data owner to immediately react to the destruction of stored data, as well as it implicitly motivates data holders to keep the data in their storage space. The latter aspect which involves remuneration incentives (rewarding cooperative nodes and punishing selfish nodes), is studied theoretically using game theory.

2. Existent protocols for verifying data possession

Protocols for the verification of data possession have been mostly studied within P2P systems, which have addressed the problem of storage or backup of some data in a distributed and self-organized fashion.

For a backup application, the verified data is still held by the owner node, therefore, verification of data possession solely engages a fresh proof of the stored data integrity. However, for a storage application, verifying the possession of data that the owner does not hold anymore cannot simply consist in checking an integrity code. Verification requires as well a proof of data origin.

In the literature, two categories of protocols can be distinguished: the first one relies on the verification of some data sent by the holder against the original data kept by its original owner, while the second one is based on the verification of a proof generated on demand (when receiving a challenge) from the data. The former category has been most used for backup applications, since data are preserved at the originator, while the latter is better suited for distributed storage applications in general. The protocol proposed in this report pertains to the latter category.

2.1. Verification against original data

The first family of verification approaches regularly challenges the data holder to send back the original data: the verification of the holder's response is then compared with the original data.

In the cooperative Internet backup scheme described in [5], each peer periodically challenges its partners by requesting a block out of the stored data. The response is checked by comparing it with the valid block stored at the verifier's disk space. This work assumes that all blocks stored at an unreliable node are probably lost if one block cannot be recovered. It also assumes a P2P connectivity model, that is, most nodes are reachable. A mobile ad-hoc backup application makes essentially inverse assumptions.

In [6], the prover has to send the MAC of data as the response to the challenge message. The data originator sends a fresh nonce (a unique and randomly chosen value) as the key for the message authentication code: this is to prevent the holder node from storing only the result of the hashing of the data.

2.2. Verification using an on-demand proof

The second type of verification protocol answers a challenge by cryptographically combining a proof of the existence of the data at the holder together with some assurance regarding the freshness of this proof.

[7] offers two solutions: the first one requires pre-computed results of challenges to be stored at the verifier, where a challenge corresponds to the hashing of the data concatenated with a random number. The size of the pre-computed information required is smaller than the data size. Compared with [5] and [6], the protocol requires less storage at the verifier, yet it allows only a fixed number of challenges to be performed. The second solution uses an RSA-based proof. This solution requires little storage at the verifier side yet makes it possible to generate an unlimited number of challenges.

A similar RSA-based solution is described by Filho and Barreto in [8] that makes use of a key-based homomorphic hash function H. In each challenge of this solution, a nonce is generated by the verifier which the prover combines with the data using H to prove the freshness of the answer. The prover's response will be compared by the verifier with a value computed over $H(data)$ only, since the secret key of the verifier allows the following operation (d for data, and r for nonce):

$$H(d+r)=H(d)\times H(r) \quad (1)$$

Both solutions in [7] and [8] however use exponential operations where the size of the exponent is approximately equal to the data size, which entails a poor performance. So as to boost the performance, the authors of [8] suggest the use of elliptic curves.

We propose in this report a verification protocol that is cheaper, in terms of CPU usage, than protocols in [7] and [8], and that does not require original data or pre-computed information to be stored at the verified side.

3. Probabilistically verifying data possession

We consider a data storage application (for instance in the general context of a mobile ad hoc network) in which a node may cooperatively store its personal data at another node, taking advantage of the excess storage space offered by the latter node. In a naive approach, it may be expected that the evaluation of the availability of data and of the cooperation of holder nodes is delayed until the data stored are accessed again or retrieved. Consequently, any reaction to the destruction of data (e.g., increasing data replication, penalizing non cooperative holder node) also only happens lately. Most approaches to distributed and especially peer-to-peer storage instead set on periodically verifying if data holder nodes still possess the data they promised to store and thereby rely on a protocol for verifying data possession.

Such a verification protocol is generally depicted as a challenge with which the data owner node (that we call O) regularly contacts the data holder node (that we call H). From these periodic verification operations, responses to challenges are a means for O to infer the behavior of H. However, this periodic validation comes at an additional communication and computational cost. We suggest to address this

concern through the use of a probabilistic verification instead of a deterministic one as do most of existent proposals. For example, if the holder misbehaves with a probability r of data destruction per regular period; the holder node misbehavior will be detected in average after $1/r$ challenges.

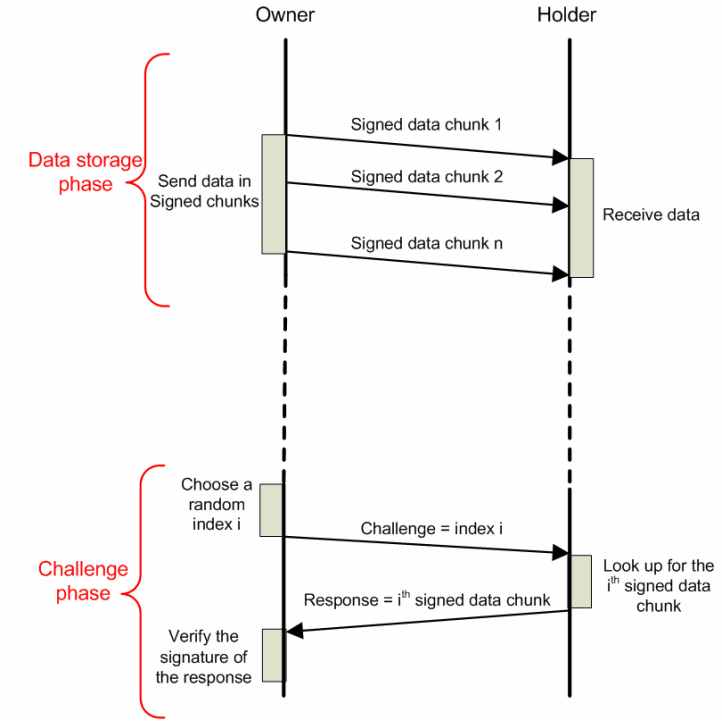

Figure 1 Data storage phase and challenge phase

The verification protocol summarized in Figure 1 (see [9]) comprises two phases:

Storage phase: The data is split into n indexed chunks, which are then encrypted. Additionally, the integrity of chunks is protected by "signing" the data: digital signature may be used for that purpose but is an expensive cryptographic mechanism; it is possible to use other less expensive methods like DES symmetric encryption [10] with a secret key, or keyed one-way hash function such as HMAC [11], or even concentric encryption. All protected chunks are sent to the holder.

Challenge phase: the owner randomly chooses one index corresponding to a data chunk (the probability to choose this index the next time does not change) and sends it to the holder. The holder answers with the corresponding chunk and its signature. The owner verifies the validity of the signature (and then, the chunk is of course deleted). The challenge operation is repeated periodically until either the owner retrieves its data or it detects that the holder has destroyed a data chunk.

The verification protocol only requires the owner node to have the public key for the signature, or the secret key if some form of hashing is used. The challenge phase only verifies one data chunk, which makes it easier to handle for environments with scarce resources.

Since chunk indexes are chosen randomly, the holder node should keep all chunks stored to answer correctly all possible challenges from the owner node.

In this protocol, the selfish node may be depicted as a node destroying a portion of data chunks, as example the holder node destroys k chunks over the n chunks, $r=k/n$. So, the holder can answer correctly to challenges with probability $(1-r)$. The number of challenges sent by the owner node per time period determines the rate at which node misbehavior may be detected.

Incentives: incentives are necessary to foster the cooperation of holder nodes in order to reduce the number of node misbehaviors (this report only considers selfish misbehavior).

Reputation is one first type of incentive: a node acquires good reputation as a recognition of a long-lasting behavior towards storage durability, and is deterred to misbehave by the potential loss of all reputation. Although reputation provides a very flexible incentive mechanism perfectly adequate for a single node, distributing behavior assessment exposes the system to two extremes, namely denial of service attacks and collusion attacks.

Remuneration incentives instead fit well the cooperation requirements of large scale peer-to-peer storage. The following sections analyze how the proposed verification protocol helps enforcing cooperation within a game theoretic model where cooperation incentives are expressed as remunerations: for a correct challenge-response, the holder node is rewarded, while for an incorrect response, the holder node pays a punishment charge.

4. Game theoretic model

The proposed verification protocol based on a challenge-response for the inspection of a holder node is modeled using the analytical framework of game theory [12]. These concepts provide a language to formulate structure, analyze, and understand strategic scenarios. In our model we assume the presence of two players (also termed actors) involved in the strategic process of deciding whether to cooperate or not on one hand, and to punish or reward on the other hand.

Our game model assumes that the network meets the following requirements: a) the owner node can communicate with and subsequently challenge the holder node throughout the storage duration; b) the forthcoming loss of connection may be detected (e.g., reduction of signal transmission power, no answer to ping, notification before node shutdown …): this makes it possible to distinguish selfishness from faults or disconnections.

4.1. Game elements

The essential elements of our model are:

- Players: the individuals who make decisions: data owner and data holder. A player is assumed "rational", i.e., a player is a participant in the game and whose goal is to choose the actions that produce his most preferred outcomes.
- Payoffs: the numeric values assigned to the outcomes produced by the various combinations of actions. Payoffs represent the preference ordering of players over the outcomes.
- Information: information set for a player summarizes what the player knows when it gets to make a decision.
- Chance: probability distribution over chance events. We represent chance events by a random move of nature which is a pseudo-player whose actions are purely mechanical and probabilistic.

4.2. Game models

The proposed distributed storage application is modeled as a Bayesian game. A Bayesian game is a game in which information about characteristics of the other players (i.e. payoffs) is incomplete. The game is modeled by introducing nature as player in the game.

Figure 2 illustrates the structure of our one-stage game in the extensive form (in the form of a tree where there is a complete description of how the game is played over time). A one-stage game corresponds to the phase of one challenge conducted by the owner node towards the holder node. Notations used in figures are explained in Table 1.

The parameters G, R, R' and S are measured in the same units, e.g., the number of data bytes or data chunks stored. Also regarding data stored in a distributed fashion, we presume that the remote storage space has more value than local storage space, which explains that $G>R>S$.

Table 1 Notations

Notations		Explication
Players	O	data owner
	H	data holder
Types	C	H is cooperative
	S	H is selfish
	F	H is faulty
Signals	*s*	succeed O's challenge
	f	fail O's challenge
Actions	*a*	reward H
	k	do not do anything
	d	punish H
Payoffs	*G*	distributed storage gained by O
	S	storage ceded by H
	R	reward charge, such that $R>S>0$
	R'	punishment charge, such that $G-R>R'>0$
Chance	*q*	probability of challenge's success for a selfish holder H

In the game (Figure 2), the holder H chooses its type. O is not informed about H's type, that's why, O can not distinguish between "C" and "S" if the signal sent by H is "s", and between "S" and "F" if the signal sent is "f". However O can probabilistically determine the type based on its prior beliefs about H's type; typically these beliefs reflect H's reputation. After observing a signal, O updates its beliefs according to Bayes' rule. The game model is a sequential game with incomplete information (O is not informed about H's type).

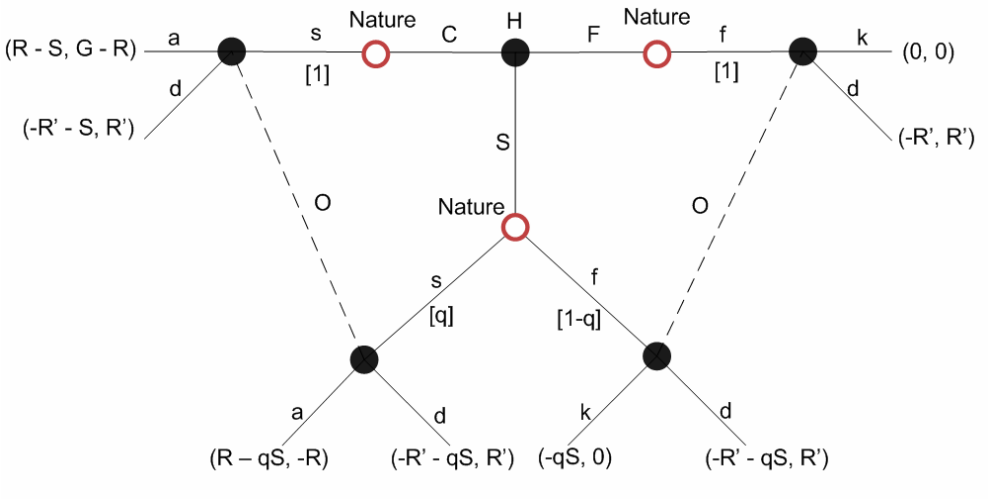

Figure 2 A Bayesian challenge-response game

4.3. Equilibriums

The solution of the game is called equilibrium which constitutes player's best response to the actions of the other player. We will define the Nash equilibrium and the perfect Bayesian equilibrium of the game in the following.

Nash Equilibrium: Nash Equilibrium is the set of players' strategy choices where no player can benefit by changing its strategy while the other player keeps its strategy unchanged. To define the Nash equilibrium of the game, the normal form of the game (which lists each player's strategies and the payoffs

that result from each possible combination of choices) is presented below in Table 2.

Table 2 Normal form game

		O		
		a	k	d
H	C	(R-S, G-R)	--	(-R'-S, R')
	S	(R-qS, -R)	(-qS, 0)	(-R'-qS, R')
	F	--	(0, 0)	(-R', R')

We assume that $G\text{-}R > R'$. If H chooses the type "C", then O, by strict dominance, chooses the action "a" because the payoff associated to "a" ($=G\text{-}R$) is higher than the payoff associated to "d" ($=R'$). By choosing "a", the better response by H is "S" because $R\text{-}S<R\text{-}qS$, and so, O will prefer to choose "d" because $R'>0>-R$. But, then, H chooses "F". At this point, neither O nor H can have a benefit by changing to another strategy. So, ("F", "d") is a Nash equilibrium. The normal form game leads to an equilibrium where non-cooperation is the best response for players.

Compared to the extensive form game, the normal form game lacks the information on whether O is informed or not about the type of H. The view of incomplete information is not represented within the normal form. Another equilibrium, the perfect Bayesian equilibrium, takes into account this view.

Perfect Bayesian Equilibrium: A perfect Bayesian equilibrium is a strategy profile $\sigma^*=(\sigma_1^*,\sigma_2^*)$ and posterior beliefs $\mu(\cdot|m)$ such that:

1. $\forall$type t, $\sigma_1^* \in \arg\max_{\sigma_1}(U_1(\sigma_1,\sigma_2^*,t))$
2. $\forall$signal m, $\sigma_2^* \in \arg\max_{\sigma_2}(\sum_t \mu(t\mid m)U_2(m,\sigma_2,t))$
3. $\mu(t\mid m)=\dfrac{p(t)\times\sigma_1^*(m\mid t)}{\sum_{t'}p(t')\times\sigma_1^*(m\mid t')}$

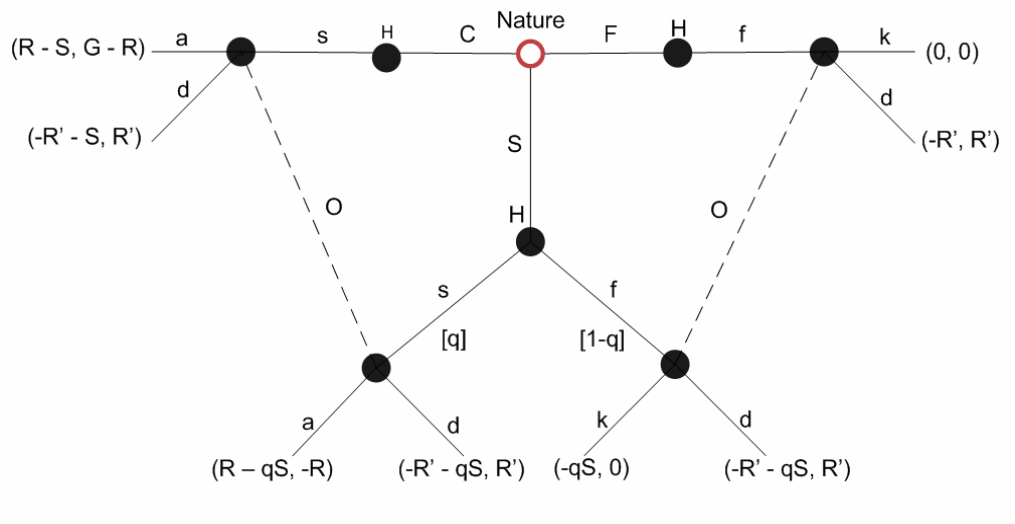

Figure 3 A signaling game

We derive a second game based on signals of the challenge-response mechanism that allows modeling a game where the owner O is certainly not informed about H's type but it has some beliefs about this type. Typically, a signaling game is a game in which an informed player gets to move first (perhaps signaling some of the information it has) and the uninformed player gets to move second, making use of the

information revealed from the first stage. The signaling game proposed (Figure 3) is similar to the game in Figure 2, the only difference resides in that H's type is chosen probabilistically.

Finding equilibrium for this game involves finding the following probabilities ([13]):

$\sigma_1^*(s \mid C) = 1 \qquad \sigma_1^*(f \mid C) = 0$

$\sigma_1^*(s \mid S) = q \qquad \sigma_1^*(f \mid S) = 1 - q$

$\sigma_1^*(s \mid F) = 0 \qquad \sigma_1^*(f \mid F) = 1$

$\sigma_2^*(a \mid s) = u_1 \qquad \sigma_2^*(k \mid s) = v_1 = 0 \qquad \sigma_2^*(d \mid s) = 1 - u_1 = w_1$

$\sigma_2^*(a \mid f) = u_2 = 0 \qquad \sigma_2^*(k \mid f) = v_2 \qquad \sigma_2^*(d \mid f) = 1 - v_2 = w_2$

Thus, the belief update equations are as follows:

$\mu(C \mid s) = p(C) / (p(C) + p(S) \times q)$

$\mu(S \mid s) = p(S) \times q / (p(C) + p(S) \times q)$

$\mu(F \mid s) = 0$

$\mu(C \mid f) = 0$

$\mu(S \mid f) = p(S) \times (1 - q) / (p(S) \times (1 - q) + p(F))$

$\mu(F \mid f) = p(F) / (p(S) \times (1 - q) + p(F))$

H's payoffs corresponding to each type is given by:

$\mathrm{U}_1(\sigma_1, \sigma_2^*,\ C) = u_1 \times (R + R') - R' - S$

$\mathrm{U}_1(\sigma_1, \sigma_2^*,\ S) = q \times [u_1 \times (R + R') + R' \times w_2 - R' - S] - R' \times w_2$

$\mathrm{U}_1(\sigma_1, \sigma_2^*,\ F) = -R' \times w_2$

Expected O's payoffs for each signal sent by H is given by:

$$\sum_t \mu(\mathrm{t} \mid \mathrm{s}) \mathrm{U}_2(s, \sigma_2,\ t) = u_1 [G p(C) / (p(C) + p(S) q) - R - R'] + R' \sum_t \mu(\mathrm{t} \mid \mathrm{f}) \mathrm{U}_2(f, \sigma_2,\ t) = R' w_2$$

There are two case solutions:

Case 1: if $G \times p(C) / (p(C) + p(S)) - R - R' \geq 0$, then σ_2* is maximized for u_1 = 1 and w_2=1. Because $R+R'-S>0$, σ_1* is maximized for q = 1. The perfect Bayesian equilibrium is the strategy where:

$\sigma_1^*(s \mid S) = 1 \qquad \sigma_1^*(f \mid S) = 0$

$\sigma_2^*(a \mid s) = 1 \qquad \sigma_2^*(k \mid s) = 0 \qquad \sigma_2^*(d \mid s) = 0$

$\sigma_2^*(a \mid f) = 0 \qquad \sigma_2^*(k \mid f) = 0 \qquad \sigma_2^*(d \mid f) = 1$

$p(S) / p(C) \leq (G - R - R') / (R + R')$

The equilibrium of the game leads to a strategy where O and H cooperate.

Case 2: if $G \times p(C) / (p(C) + p(S)) - R - R' < 0$, then σ_2* is maximized for w_2=1 only. The choice of u_1 is dependent on q and vice versa. If u_1 = 0, then σ_1 is maximal for q = 0, and for q = 0, σ_2 is maximal for u_1 = 1, and for u_1 = 1, σ_1 is maximal for q = 1, however, for q = 1, σ_2 is maximal for u_1 = 0, and so on. There is no perfect Bayesian equilibrium for this case.

5. Repeated game

We analyze a class of repeated games in which the informed player's type is persistent and the history of actions is perfectly observable. This context rightly represents the periodic iteration of the verification protocol performed by the owner node to assess whether the holder node is still storing the data it promises to store. The analyzed repeated game is the game of Figure 2 and Figure 3 iterated while maintaining H's type. These games are played for finite times, but no player knows the exact game

termination time. The probability p captures the probability of "natural" termination of the repeated game (e.g., loss of connection between O and H). Additionally, the owner node O has the possibility to stop the repeated game if it detects the selfishness or the failure of H (H is of type "S" or "F"). The payoff at the i^{th} period is designated by $g_i=(g_i^H, g_i^O)$. The sum of per-period payoffs is given by:

$$g = (\sum_{i=0}^{\infty}(1-p)^i g_i^H, \sum_{i=0}^{\infty}(1-p)^i g_i^O) .$$

5.1. Action profiles

From the signals sent per-period by H, O may infer the type of H. There are three distinct possible action profiles:

1. (s, a), (s, a), (s, a), …
2. (f, d)
3. (s, a), (s, a), …, (s, a), (f, d)

At the first round, if the signal is "f", O infers that the type of H is either "S" or "F", for both cases it is better to play the action "d". If the signal is "s", then, the best response of O is to play "a". If the signal changes to "f", O concludes that H is of type "S" and the action played is "d".

5.2. Identification of H's type by O

The iteration of the game of Figure 2 or Figure 3 allows the owner to infer the type of H. If H is of type "F", O receives the signal "f". If H is of type "C", O receives the signal "s" during the whole repeated game. O may possibly detect the misbehavior of the holder, because by iteration, if the type of H is "S", H risks failing to a challenge with probability (1-q). The probability that the selfishness of H is detected at the l^{th} stage of the game, $p_{det}(l)$, is computed in equation (2), and the average value of this number l is given in equation (3) ((3) is applicable for $(q, p)\neq(1, 0)$, otherwise $E[l]$ is infinite)

$$p_{\det}(l) = q^{l-1}(1-q)(1-p)^{l-1} \tag{2}$$

$$E[l] = \sum_{l=1}^{\infty} l \times q^{l-1}(1-q)(1-p)^{l-1} = (1-q)/(1-q(1-p))^2 \tag{3}$$

If the holder is of type "S" (selfish), it must choose a probability of success q that augments its chances of not being detected by O, so that it is rewarded several times before O detects its selfishness and punishes it, i.e., $E[l]$ must be maximized. The value q^*=1-[p/(1-p)] corresponds to the maximum of $E[l]$. This is valid only for p<0.5. For the cases where p>0.5, $E[l]$ is maximum for q=0.

5.3. Requirements on incentive charges

Assumptions of Figure 2 state that *G-R>R'*, and *R>S*. It is possible to deduce likewise requirements from the repeated game over the incentive charges R and R'. We denote the probabilistic gain of H if it chooses to be cooperative (respectively selfish) as $g^H_{type=C}$ (respectively $g^H_{type=S}$). Similarly, gains obtained for O, when H is cooperative, and O chooses "a" (respectively "d") are denoted $g^O_{type=C,a}$ (respectively $g^O_{type=C,d}$). The gains of the holder and the owner in these cases are computed.

$$g^H_{type=C} = (R-S)/p$$

$$g^H_{type=S} = \frac{q \times R + (1-q) \times (-R') - q \times S}{1 - q \times (1-p)}$$

$$g^O_{type=C,a} = (G-R)/p$$

$$g^O_{type=C,d} = R'$$

H is more favorable to the cooperative strategy than to the selfish strategy, if $g^{H}_{type=C} > g^{H}_{type=S} \Leftrightarrow R > S - p \times R'$. And, O is strategically more interested to cooperate with H, if $g^{O}_{type=C,a} > g^{O}_{type=C,d} \Leftrightarrow G > p \times R' + R$.

5.4. Numerical evaluation of the game framework

The games of Figure 2 and Figure 3 are iterated and evaluated within different scenarios. The evaluation is performed using a custom simulator, and games' parameters are measured in MB (Mega Bytes) unit (1 MB=10^6 bytes).

The evaluated scenarios permit to define additional requirements on the values of the reward returned *R*, the punishment charge *R'*, and the impact of the probability *q* and the probability *p* on the cooperativeness of the holder.

At first, we consider the repeated game of Figure 2. Payoffs of the holder are collected in Figure 4 varying the reward *R* for different storage costs *S*. H chooses the strategy that maximizes its payoff. To make H choose the type "C" over "S", its outcome by choosing "C" must be higher than its outcome choosing "S". For this, the reward *R* must be bigger than a minimum value (for example, for *S*=10, *R* must be bigger than 11). So, *R* must verify a minimum threshold to motivate the cooperation of H (Figure 6).

For the chosen value of the reward *R*, the punishment charge *R'* must verify as well a different minimum value to make H chooses the cooperative strategy (Figure 5). For example, for *R*=11, *R'* must be bigger than 1, and for *R*=20, all values of *R'* between 0 and (*G*-*R*) are permitted. Thus, it is possible to confine the incentives to rewarding the cooperative player. But then, the minimum threshold for the reward R increases (see Figure 6).

For the specified values assigned to the game parameters (*G*=30, *R*=20, *R'*=5, *S*=10, *p*=0.2, *q*=0.5), Figure **7** shows the average outcomes per-period collected for each player when H chooses the strategies "C" (cooperative), or "S" (selfish), or "F" (faulty). For the first round, H's outcome being selfish exceeds H's outcome being cooperative. But this changes considerably after the first round. The total payoff during the repeated game turns the profit to the cooperative strategy. All (average) outcomes decrease over time because they occur with probability 1-*p* (=0.8). The figure demonstrates that the best strategy for H is the cooperative strategy, given these appropriately-chosen game parameters.

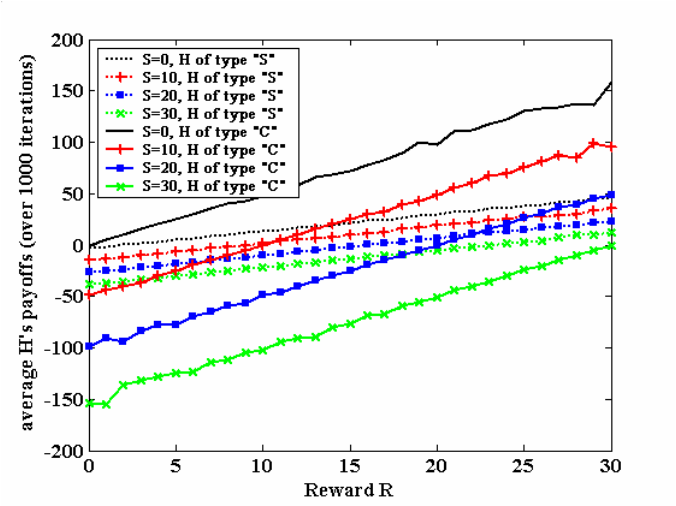

Figure 4 Average H's payoff over the reward value *R*, varying storage cost *S*. *G*=30, *R'*=5, *p*=0.2, *q*=0.5.

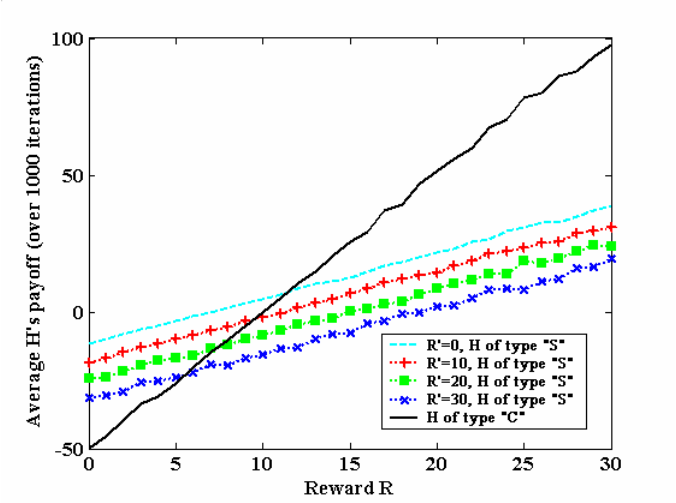

Figure 5 Average H's payoff over the reward value R, varying punishment charge R'. G=30, S=10, p=0.2, q=0.5.

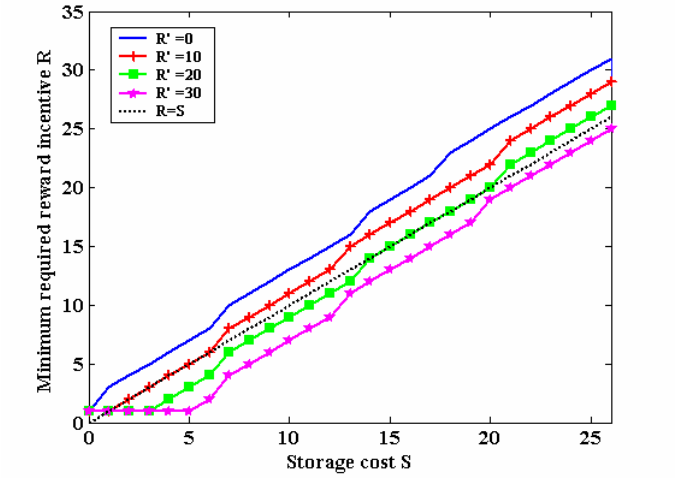

Figure 6 Minimum threshold for the reward R function of the storage cost S, for various values of punishment incentive R'. G=30, p=0.2, q=0.5.

The beneficial impact for H when increasing the probability q (p is low) is illustrated in Figure 8. Increasing q has the opposite impact on O's payoffs (because even if q approximates 1, O will not benefit from the portion of data stored in H). The variation due to different values of p is less significant for H than for O, this is because the increase of H's payoff when q increases is restrained by the storage cost (qS) that becomes more important. Besides, we notice that for p roughly bigger than 0.6 (Figure 9), H must strategically choose to be selfish with a ratio q very low (=0.1), however, for p less than 0.6, a high ratio q (=0.9) is better for H. We have come to this result numerically in II-C-2, although the point of curve change was at p=0.5 (the statistics in II-C-2 did not take into account the storage cost qS that increases with q). This demonstrates that the repetition of the game motivates the holder H to cooperate (q approximating 1).

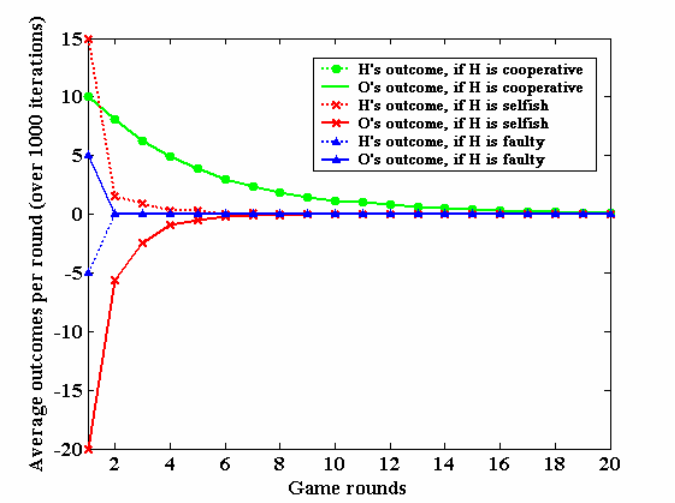

Figure 7 Average O and H's outcomes per round during the repeated game. G=30, R=20, R'=5, S=10, p=0.2, q=0.5.

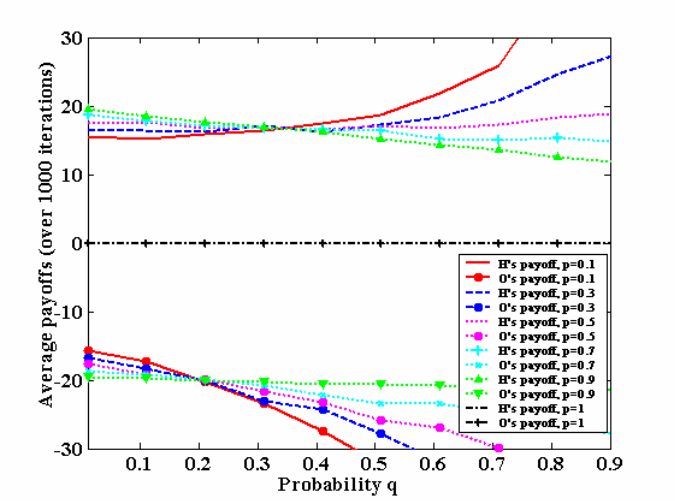

Figure 8 Average O and H payoffs if H is "selfish" over the probability q, for various values of p. G=30, R=20, R'=5, S=10.

At this point, we consider, from now and on, the repeated game of Figure 3. We are interested for this repeated game in defining the probability q for which H maximizes its payoff, and also to identify the cases in prior beliefs where O has better to defect.

Figure 10 shows again the good outcome of H being cooperative since H's payoff significantly increases when p(C) increases. The figure shows as well that the most beneficial strategy for H, if it is of type "S", is to choose a high probability q (q approximating 1).

In the previously studied one stage signaling game, we put forward the inequality $p(S)/p(C) \le (G-R-R')/(R+R')$, for which we have a perfect Bayesian equilibrium where both O and H cooperates. However for the repeated game, this inequality does not exactly hold. Figure 11 demonstrates that there exists a given ratio of prior beliefs p(S)/p(C) such that above this ratio O must not cooperate (i.e., O must stop the game by playing the action "d", for punishment). For example, for the given particular game' parameters, if p(S)/p(C)=4 (e.g., p(S)=0.4 and p(C)=0.1), O must play "d" because its average payoff if it plays other than "d" is less than R'=5.

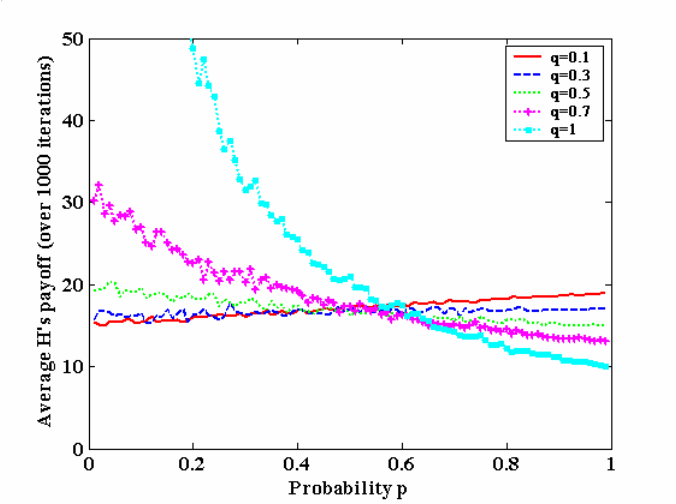

Figure 9 Average H payoffs if H is "selfish" over the probability p, for various values of q. G=30, R=20, R'=5, S=10.

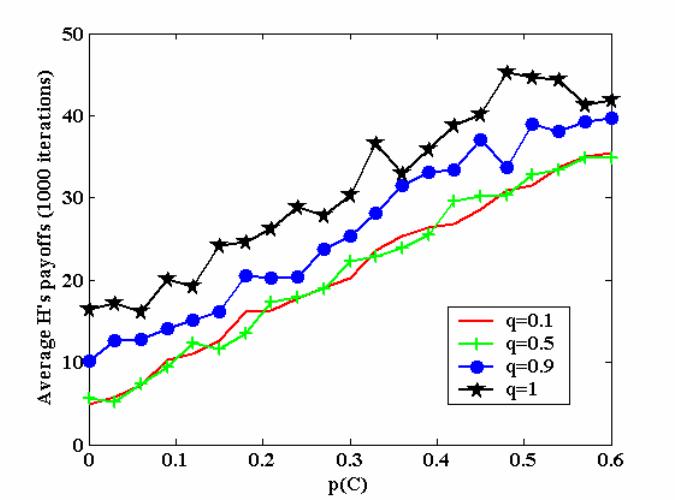

Figure 10 Average H's payoffs over prior beliefs on the type "C", varying the probability of success q. G=30, R=20, R'=5, S=10, p=0.2, p(S)=0.3.

Summary

The repeated game of Figure 2 represents an interaction between a data owner and a data holder from a data holder perspective. For this repeated game, we aim to encourage the cooperation of the holder by making its cooperative behavior the best strategically choice to make. For this, we showed the inequalities that the reward R and the punishment R' should verify. We demonstrated as well that it is possible to restrict the incentives to simply rewarding the holder (R'=0). Besides, the result on the probability p shows that iteration of the game favors the cooperativeness of H. On the other hand, the repeated game of Figure 3 illustrates the interaction of a data owner with a holder from the owner perspective. For this repeated game, we aim, this time, to guide the owner in choosing the best response to holder actions based on the prior beliefs about this very holder. These prior beliefs correspond to holder reputation. Using numerical results, it is possible to define which actions the owner must follow for a given ratio p(S)/p(C).

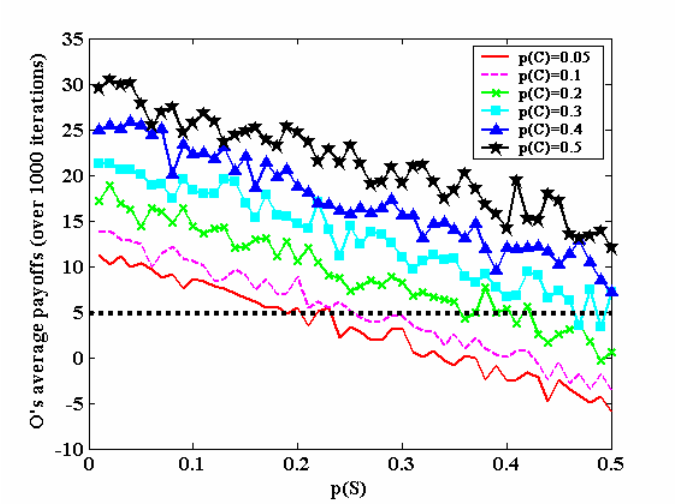

Figure 11 Average O's payoffs over prior beliefs on the type "S", varying the prior beliefs over the type "C". G=30, R=20, R'=5, S=10, p=0.2, q=0.5.

6. Conclusion

In this report we proposed a verification protocol as a means to construct a periodic cooperation detection mechanism in the context of peer-to-peer distributed storage for wireless ad hoc networks. The protocol was designed to indirectly motivate nodes to behave cooperatively. The inherent incentive compatibility property of the proposed protocol was theoretically validated using a game theoretical model. We showed that the Bayesian game model of our protocol allows solutions where both parties of the game are cooperative.

As future work, we plan to construct the whole mechanism of cooperation enforcement, and to validate the mechanism by means of realistic simulation scenarios and results obtained from a real experimentation.

7. References

[1] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In Proceedings of HotOS VIII, May 2001.

[2] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An architecture for globalscale persistent storage. In Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Nov. 2000.

[3] R. Dingledine. The free haven project: Design and deployment of an anonymous secure data haven. Master's thesis, MIT, June 2000.

[4] Adya et al. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. Usenix OSDI, 2002.

[5] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme", *In Proceedings of the 2003 Usenix Annual Technical Conference (General Track), pp. 29-41*, San Antonio, Texas, June 2003.

[6] G. Caronni and M. Waldvogel. Establishing Trust in Distributed Storage Providers. In *Third IEEE P2P Conference*, Linkoping 03, 2003.

[7] Y. Deswarte, J.-J. Quisquater, and A. Saïdane. Remote Integrity Checking. *In Proceedings of Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS)*, 2004.

[8] D. G. Filho, P. S. L. M. Barreto. Demonstrating data possession and uncheatable data transfer. In *IACR Cryptology ePrint Archive*, 2006.
[9] N. Oualha and Y. Roudier. Probabilistically secure cooperative distributed storage. Technical Report RR-07-188, Institut Eurécom, Feb 2007.
[10] National Bureau of Standards. Data Encryption Standard. *Federal Information Processing Standards Publication No. 46*, , January 15, 1977.
[11] M. Bellare, R. Canetti, and H. Krawczyk. HMAC: Keyed-Hashing for Message Authentication. *RFC 2104*, Internet Engineering Task Force, February 1997.
[12] Theodore L. Turocy and Bernhard von Stengel. Game theory. *Cdamresearch report lse-cdam-2001-09*, London School of Economics, Oct 2001.
[13] Farhad Ghassemi. Signaling games, 2006. Available: http://www.cs.ubc.ca/~kevinlb/teaching/cs532a%20-%202006/Projects/FarhadGhassemi.pdf.