UNIVERSITÉ DE CERGY-PONTOISE

UNIVERSITÉ DE PARIS X - NANTERRE

Repeated games played by cryptographically sophisticated players

Olivier Gossner*

N° 99-07

Thema

Théorie Economique, Modélisation et Applications

UMR 7536 - Centre National de la Recherche Scientifique

CERGY - PONTOISE Université

UNIVERSITE Paris X NANTERRE

Repeated games played by cryptographically sophisticated players

Olivier Gossner*

N° 99-07

February, 1999

The author is grateful to the members of the GRECC for their initiation to cryptography, and particularly to Jacques Stern and Serge Vaudenay. Insightful comments from Nathan Linial and Jean-François Mertens are gratefully acknowledged. The author also benefited from many conversations with Gaetano Bloise, Francis Kramarz, Igal Milchtaich and Philippe Riviere. The author is the only responsible for all remaining errors.
Part of this research was done while the author was visiting CORE, Université Catholique de Louvain.

* THEMA (UMR CNRS 7536), Université Paris X-Nanterre

Abstract:

One of the main goals of bounded rationality models is to understand the limitations of agent's abilities in building representations of strategic situations as maximization problems and in solving these problems. Modern cryptography relies on the assumption that agents's computations should be implementable by polynomial Turing machines and on the existence of a trapdoor function. Under those assumptions, we prove that every correlated equilibrium of the original infinitely repeated game can be implemented through public communication only.

Résumé :

Un des objectifs fondamentaux des modèles de rationalité limitée est de comprendre les limitations des agents dans leurs capacités à représenter des situations stratégiques sous forme de problèmes de maximisation et dans leurs aptitudes à résoudre ces problèmes. La cryptographie moderne impose que les stratégies des agents doivent être réalisables par des machines de Turing en temps polynomial et suppose l'existence d'une *fonction trappe*. Nous montrons que sous ces hypothèses, tout paiement d'équilibre corrélé d'un jeu infiniment répété est un paiement d'équilibre en communication publique sous forme extensive du jeu répété.

1 Introduction

One of the main goals of bounded rationality models is to understand the limitations of agent's abilities in building representations of strategic situations as maximization problems and in solving these problems. Modern cryptography relies on the assumption that agents's computations should be implementable by polynomial Turing machines. The aim of this paper is to study the implications of such a limitation on agents' computational powers when applied to repeated games with public communication.

The central issue of cryptography is security of communication. Two agents A and B exchange messages while C tries to spy them. Usually, no secure channel is assumed so that any message can be intercepted by C. To counter this, A and B code their messages so that C can not understand them. To send the information x to B, A uses a key k_e (the encoding key) to encode x into $y = k_e(x)$ and sends message y to B. After receiving y, B uses a key k_d (the decoding key) to compute $z = k_d(y)$. The pair of keys (k_d, k_e) is chosen such that [1] $z = x$ for all x and [2] C cannot get information on x from knowledge of y without possessing information on k_d.

Classical cryptography assumes that A and B share some secret information before they start to exchange messages. One can imagine that they were in the same location where they agreed on a pair (k_e, k_d) without being spied by C. These keys can be used to encode and decode messages when A and B are in separate locations. For instance, k_d and k_e may be formed of the same sequence of l alphabet characters. To code a message x consisting of no more than l alphabet characters, A constructs y from x by adding (modulo 26) the i-th character of k_e to the i-th character of x. B can easily do the reverse operation to retrieve x. An unfortunate consequence of Shannon's information theory [Sha48] to such classical cryptosystems is that the length (or more precisely the entropy) of x cannot exceed the length of the keys, otherwise conditions [1] and [2] cannot hold together. This limitation on

the size of x can be easily seen in our example. In general, all such codes can be broken using appropriate statistical tools on long enough records of communication between A and B.

Modern cryptography, as initiated by Diffie and Hellman [DH76], proposes ways for A and B to communicate secretly for as long as they wish even if they do not share any secret information to start with. In typical modern cryptosystems, B starts by choosing randomly a decoding key k_d, then computes an encoding key k_e and sends it to A (or even announces it publicly). After this, messages are then sent from A to B using the pair of keys (k_e, k_d). We can even assume there exists a one-to-one mapping from k_d to k_e. So, one may wonder why it is that C cannot simply compute k_d from k_e, and x from k_d and y. the answer to this question lies in the boundedness of agent's rationality. All the computations needed by A and B can be done in reasonable time, whereas computing k_d from k_e would take ages. Cryptosystems used in everyday life are such that one can compute k_e from k_d, y from k_e and x from y and k_d in a few seconds with a personal computer, whereas getting k_d from k_e would take more than a century using the best known algorithm on the most powerful existing machines. The function used by A to send a message to B is called a trapdoor function: it is easy to compute using k_e, but hard to invert without knowing the trapdoor information k_d.

Modern cryptography relies on two assumptions. The first assumption is that agents should be represented by polynomial Turing machines. The second assumption is the existence of a trapdoor function such as described above. As we wish to adopt these assumptions in a game theoretic set-up, let us now discuss them.

Turing machines is the model of computation commonly used by computer scientists, each Turing machine being the implementation of an algorithm. Consider a Turing machine that inputs the description of a task to perform associated with some integer representing the degree of complexity of

that task. This machine is called polynomial when the number of operations needed for the computations is bounded by some polynomial of the degree of complexity. Otherwise it is called non polynomial. Modern cryptography assumes that agents must be represented by polynomial Turing machines.

As a bounded rationality model, it may be useful to situate polynomial Turing machines in perspective with automata and with general Turing machines. If fact, every automaton can be seen as a particular polynomial Turing machine (since an automaton is fully described by its transition function it needs just one operation for every computation), and every polynomial Turing machine is by definition a Turing machine.

Automata were widely developed as a model of bounded rationality in repeated games (Aumann, [Aum81], Rubinstein [Rub86], Neyman [Ney85] [Ney98], Ben Porath [BP93]). This model is well fit to describe simple behavior rules like in evolutionary game theory where the behavior of agents is seen as a consequence of their genes, or of their memes (Axelrod [Axe84], Binmore and Samuelson [BS92]). The main characteristic of an automaton, namely its number of states, also provides a useful measure of the complexity of the strategy implemented by this automaton. On the other hand, an automaton cannot count up to a number greater than its number of states, and therefore cannot keep track of a calendar indefinitely. As far as complexity is concerned, keeping track of a calendar is a rather easy task for an human being which cannot be performed by an automaton.

The reason why the memory of an automaton is bounded is that such machines do not possess any ability to read or to write. Turing machines can be simply described as automata which are provided the ability to read and write on potentially infinite tapes. Turing machines form a wide model of rationality since –even though such a statement can never be proved formally–, Church's thesis asserts that they can implement any algorithm one may ever construct. Yet, some problems exist that cannot be solved by any Turing machine, and which are called undecidable. A wider class of problems are

those that cannot be solved by any polynomial Turing machine, and which are called intractable. No algorithm can solve undecidable problems, whereas intractable problems are just not solvable in practice. Binmore [Bin87] [Bin88] advocates Turing machines as a model for the inductive reasoning process through which agents take decisions. He argues that such a process should essentially be algorithmic, and quotes Megiddo's [Meg86] oxymoron: "... a fully rational player...can even decide undecidable problems".

We wish to argue that if players cannot solve undecidable problems because of the structure of their logic reasoning, natural time limitations should also imply that they are not able to solve intractable problems either. Hence we shall restrict the strategy sets available to the players to polynomial Turing machines.

We may now briefly discuss our second assumption, namely the existence of a trapdoor function, which is a function which is easy to compute, but hard to invert without the knowledge of some "trapdoor" information. Mathematically speaking, the existence of such a function has never been proved and is still an open question. From a practical point of view, there exist some functions which are empirically trapdoor functions in the sense that they are easy to compute and despite of many attempts, no way of inverting them without knowledge of the trapdoor information has been found. Important methods of secure payment by credit card or through the internet such as the RSA cryptosystem rely on the fact that no-one knows how to invert those functions in a practical way.

We study a model of infinitely non-discounted repeated games with public communication at each stage. Our result is that any correlated equilibrium payoff of the "classical" infinitely repeated game obtains as an equilibrium payoff of our model if players are represented by polynomal Turing machines. For games with incomplete information with at least 4 players, Forges [For90] proved that any communication equilibrium payoff (and hence any correlated equilibrium payoff) of the repeated game with no communication could be

implemented by an equilibrium of the repeated game in which players communicate through private phone lines between each two consecutive turns. Here, considering games of perfect information, we obtain a similar result when only public communication is allowed. Situations with public communication include cases in which players communicate through radio or television networks, and more generally cases in which no group of players can exchange messages while making sure no-one can spy their conversation. We thus prove that all correlation possibilities can be implemented using public communication only. Were more means of communication available to the players, the set of equilibrium payoffs would remain unchanged.

From a positive point of view, the "classical" Folk Theorem (Aumann and Shapley [AS94] and Rubinstein [Rub94]) is sometimes criticized as being a disappointing result. In fact, it states that the set of equilibrium payoffs is the set of payoffs that are feasible and which gives to every player at least the worst payoff the other players can force him to get by using *mixed strategies*. In our model, every payoff that is feasible and which gives to each player at least the worst payoff that other players can force him to get by using *correlated strategies* is an equilibrium payoff. We have thus augmented the underterminacy raised by the Folk Theorem. One way to get out of this pit can be the use of cooperative game theoretic concepts, as proposed by Mertens [Mer80]. For such a concept to make sense, one must assume that each coalition of players can randomize between its strategies, which is equivalent to assume that each coalition can use a public correlation device in the original game. Our work provides a foundation of how such correlation devices can be endogenously built by the players even if no private communication is possible.

In order to prove our result, we shall establish that secure communication between any group of agents can be implemented through public communication. We first extend the notion of a trapdoor function (that allows some agent A to send a secret message to another player B) into k-trapdoor func-

tions (which allow A to send a secret message separately to any group of k other players). We then prove that any trapdoor function is also a k-trapdoor function.

Finally, let us mention that the only applications so far of modern cryptography to game theory are due to Urbano and Vila [UV97] [UV98b] [UV98a]. Assuming sequential communication between two players, they propose protocols based on modern cryptography by which the players can replace any mediator of a communication equilibrium. As opposed to them, the strategies we exhibit do not rely on a particular cryptosystem, but rather on the theoretical assumption of the existence of such a cryptosystem. In fact, although some cryotpsystems have resisted all attempts by mathematicians and computer scientists to break them, none has been mathematically been proved inviolable.

We introduce and discuss an example in Section 2. Then, we introduce a model for repeated games played by polynomial Turing machines in Section 3 and Section 4. We present trapdoor functions Section 5. Our main Theorem is stated and proved in Section 6, using a generalization of trapdoor functions into k-trapdoor functions. Section 7 presents an extension to subgame perfect equilibria. We finally provide some concluding remarks in Section 8.

2 A secret rendez-vous game

Consider a 3-players game in which each player A, B, and C, chooses an action l_A, l_B, or l_C from a finite set of cardinal L. These actions have to be interpreted as different locations where players may go. The payoffs to players A and B are the same and take the value 1 if $l_A = l_B \neq l_C$, and 0 otherwise. The payoff of player C is the opposite of the payoff of A and B. Hence, the goal of A and B is to meet without the presence of C whereas C tries to prevent them from doing so.

What is the best payoff that A and B can guarantee in such a game

? This will depend crucially on the assumptions we make concerning the communication possibilities between A and B.

- Assume A and B have no mean to communicate. Then the best they can do is to randomize evenly their actions among L, thus guaranteeing $\underline{u}=\frac{L-1}{L^2}$.
- Assume A and B can talk through a private phone line. Then they can perfectly correlate their actions, and guarantee a payoff of $\bar{u}=\frac{L-1}{L}$.
- Now, assume A and B can exchange public messages, and there is no bound on the computational power of C. Then A and B cannot do better than in the first case, and guarantee at most $\underline{u}$.
- Finally, assume A and B can exchange public messages, and use a public key cryptosystem that cannot be broken by C. In this case they can exchange secret messages, and guarantee $\bar{u}$.

Note finally that for L large, $\underline{u}$ goes to 0 whereas $\bar{u}$ goes to 1, so that assuming public key cryptography may dramatically change the outcomes of the game.

3 Turing machines

A Turing machine **M** works with some input tapes, a computation tape, and an output tape. Each tape is divided in an infinite stream of cells, each of which may contain a symbol. Tapes are scanned by tape heads, which can read the symbol in the cell currently scanned, and write on the computation tape and on the output tape. **M** is controlled by a finite state control, which operates in discrete steps. At each step it performs three functions in a way dependent on its current state and on the tape symbols currently read by the tape heads:

1 - Put the unit control in a new state.

2 - Write new symbols in the tape cells currently scanned of the computation tape and of the output tape, replacing the symbols already there.

3 - Move each tape head one cell to the right, one cell to the left, or leave it where it is.

Formally, a Turing machine **M** consists of:

- A finite set of **input tapes** IT, a **computation tape** CT, and an output tape OT. The set of tapes is $T = IT \cup \{CT\} \cup \{OT\}$.
- For each $b \in T$, a finite list of **symbols** S_b. The intersection $\cap_b S_b$ contains a symbol # called the **blank symbol**.
- A finite list of **states** Q, $q_0 \in Q$ is the **initial state**, and $h \in Q$ is the **halt state**.
- A **transition function**, which is a mapping:

$$\delta : \ Q \times \Pi_{b \in T} S_b \rightarrow Q \times \Pi_{b \in T} \{L, R, S\} \times S_{CT} \times S_{OT}$$

  That is, for some state q and for some symbols read, the specification of a new state and, for each tape, a direction for the corresponding head that may be left (L), right (R), or stationary (S), and for the computation tape and the output tape, a new symbol to be written on the tape.

Initially, the machine is in state q_0, all heads are positioned to the leftmost squares of their tapes, and each tape b contains a stream of elements of S_b. From an initial configuration, the machine performs a certain number of operations, and eventually may reach state h.

It is not obvious from the previous definition what sophisticated operations can be performed by these machines, what functions they could compute, or what strategies they could play in a game. Actually, every algorithm

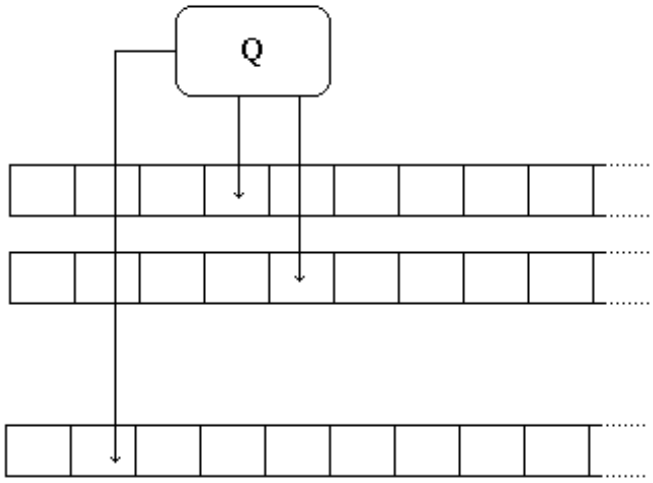

Figure 1: head and tapes of a Turing machine

for computation one may think of can be implemented by a Turing machine. This result constitutes Church's thesis[1].

The model for Turing machines we presented is not the only possible. For instance, one could assume several computation tapes, or allow the tapes to be infinite in both directions, or allow several head tapes on each tape. Nevertheless, all these models would be equivalent to this one. For more details on the various models of Turing machines and their equivalence, the reader is referred to the books of Lewis and Papadimitriou [LP81], or Aho, Hopcroft and Ullman [AHU74].

3.1 Probabilistic Turing machines

Turing machines, as defined in previous paragraph, do not have the power to randomize in their computations. We call them deterministic Turing machines. A probabilistic Turing machine is defined the same way as above

[1]Note that it is yet not true that any function can be computed by a Turing machine.

except that we allow for stochastic transition functions $\delta : \ Q \times \Pi_{b\in T} S_b \rightarrow \Delta(Q \times \Pi_{b\in T}\{L, R, S\} \times S_{CT} \times S_{OT})$.

Note that this model is not equivalent to the classical one which defines a probabilistic (also called non-deterministic) Turing machine as a machine that can read an extra tape containing an infinite stream of *i.i.d.* realisations of uniformly distributed random variables in $\{0, 1\}$. Our model rather assumes that this extra tape should contin any infinite stream of *i.i.d.* realisations of random variables, the distribution of which would be part of the definition of the Turing machine. By not restricting the probablity of the random variables players can access, our model allows permits to implement any behavioral strategy which is computable in polynomial time.

3.2 Degree of complexity

All Turing machines considered have an input tape in which some integer t, called **degree of complexity**, is coded in unary. This tape contains t times a given non-blank symbol, followed by an infinite stream of blank symbols.

3.3 Polynomial Turing machines

Let $\tilde{s} = (\tilde{s}_b)_{b\in IT}$ denote the streams of symbols initially written on the input tapes of a Turing machine $\mathbf{M}$. The number of operations performed by $\mathbf{M}$ before it reaches h if the initial state is $\tilde{s}$ is denoted $NO(\mathbf{M}, \tilde{s})$. It is an integer, or takes the value ∞ if $\mathbf{M}$ never halts.

Definition 1 *A Turing machine* $\mathbf{M}$ *is* ***polynomially bounded*** *if there exists a polynomial* P *such that:*

$$\forall \tilde{s} \in IT, \quad NO(\mathbf{M}, \tilde{s}) \leq P(t)$$

If $\mathbf{M}$ *is probabilistic, we require this inequality to hold for each* $\tilde{s}$ *with probability one.*

3.4 Turing machines working in series

Consider a series of (possibly probabilistic) Turing machines $\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^n$ such that $\mathbf{M}^k$ inputs t and $x^{k-1} \in X^{k-1}$, and outputs $x^k \in X^k$. Assume that the set X^0 is endowed with probability π. Then t and $\mathbf{M}^k$ define a transition probability from X^{k-1} to X^k (which is actually deterministic whenever $\mathbf{M}^k$ is so), so that $t\ \pi, \mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^n$ define a probability $P_{t,\pi,\mathbf{M}^1,\mathbf{M}^2,\ldots,\mathbf{M}^n}$ on $X^0 \times X^1 \ldots \times X^n$. When no confusion is possible we shall simply denote it P_t.

4 Turing machines playing repeated games

4.1 The one-shot game

$N = \{1, \ldots, N\}$ is the set of players. For any collection of sets $(Z^i)_{i \in N}$, Z and Z^{-i} represent $\Pi_{i \in N} Z^i$ and $\Pi_{j \neq i} Z^j$. $G = ((A^i)_i, g)$ is a game in which player i's finite set of actions is A^i, and with payoff function $g : A \to I\!R^N$. A mixed move for player i is a distribution over A^i, element of $S^i = \Delta(A^i)$. g is multilinearly extended to S.

The set of **feasible** payoffs is $F = \operatorname{co} g(A)$. The $\min\max$ $\overline{v}^i$ of player i is the worst payoff players different from i can force i to get when using mixed strategies:

$$\overline{v}^i = \min_{s^{-i} \in S^{-i}} \max_{s^i \in S^i} g^i(s^{-i}, s^i)$$

$IR = \{x \in I\!R^N, \forall i\ x^i \geq \overline{v}^i\}$ denotes the set of **individually rational** payoffs. According to the "Folk Theorem", the set of (possibly perfect) equilibrium payoffs of the infinite repetition of G with no discounting is $F \cap IR$.

The correlated $\min\max$ $\underline{v}^i$ of i is the worst payoff players different from i can force i to get when using correlated strategies. From the $\min\max$ theorem, it is also the best i can guarantee against the other players:

$$\underline{v}^i = \min_{\beta \in \Delta(A^{-i})} \max_{s^i \in S^i} \mathbf{E}_\beta \, g^i(a^{-i}, s^i) \quad (1)$$

$$= \max_{s^i \in S^i} \min_{s^{-i} \in S^{-i}} g^i(s^{-i}, s^i) \quad (2)$$

where $\mathbf{E}_\beta$ is the expectation operator under β. We select $\delta_{-i} \in \Delta(A^{-i})$ where the minimum is attained in (1) and $\underline{s}^i$ under which the maximum in (2) is attained.

$CIR = \{x \in I\!R^N, \forall i \; x^i \geq \underline{v}^i\}$ denotes the set of **correlated individually rational** payoffs. Note that S^{-i} can be identified to a subset of $\Delta(A^{-i})$, so that $\underline{v}^i \leq \overline{v}^i$, and $IR \subset CIR$. Recall that the set of (possibly perfect) correlated equilibrium payoffs of the infinite repetition of G with no discounting is $F \cap CIR$.

4.2 Repeated games

In our model, each player's strategy is implemented by a probabilistic Turing machine $\mathbf{M}^i$ which has access to the following tapes:

- The Time tape, denoted Time;
- A state tape for player i, denoted State(i);
- i's output tape, denoted Output(i);
- j's output tape for $j \neq i$, denoted Output(j).

Time is the tape on which at stage t the number t is coded in unary. Between any two stages of the repeated game, an extra Turing machine called the clock machine updates Time. The integer t also serves as the degree for all Turing machines.

The state tape of i contains private information of i, only i may read or write on this tape. It allows i to keep track of some information on the history

of the game. These tapes contain only blank symbols at the beginning of the game.

The output tapes are used by the machines to declare their actions and to announce their public messages. We assume there are a fixed and finite number of symbols at the beginning of these tapes used to code the action, and that the rest of the tapes are used to code the public message. Hence, the set of public messages for any player is countable.

Stage t of the repeated game is divided in an action phase and an observation phase. Each player first computes his actions and messages secretly during the action phase. After this, actions and messages are publicly observed during the observation phase. Then, next stage proceeds.

During the action phase of stage t, the Turing machine $\mathbf{M}^i$ of player i may read or write on Output(i), but cannot read or write on Output(j) if $j \neq i$. When all machines have ended the action phase comes the observation phase. During this phase, $\mathbf{M}^i$ can read Output(j) for all j, but cannot write on Output(i). At the end of the observation phase, the clock machine writes $t+1$ instead of t on Time, and stage $t+1$ starts.

The following table summarizes which tapes $\mathbf{M}^i$ may read (R) or write on (W) during the different phases:

	Action	Observation
Time	R	R
CT(i)	R/W	R/W
OT(i)	R/W	R
OT($j \neq i$)	—	R

We want to ensure that the Turing machines used by the players halt in a number of operations that does not increase too fast with the stage of the repeated game, and yet allow for a wide set of strategies. Let PA^i be the set of polynomial Turing machines that use the tapes as described above.

Any N-tuple $\mathbf{M} = (\mathbf{M}^i)_i$ of polynomial Turing machines induce for every t a probability over the sequences of actions $(a_1, \ldots, a_t)$ of length t endowed with the discrete σ-algebra $\mathcal{A}_t$, and thus a probability $P_{\mathbf{M}}$ on the plays $(a_t)_t \in A^{I\!N}$ endowed with the $\sigma-$algebra $\vee_t \mathcal{A}_t$. Let $\mathbf{E}_{\mathbf{M}}$ denotes the expectation operator under $P_{\mathbf{M}}$. The payoff induced by $\mathbf{M}$ in G_∞ is taken as $\tilde{g}^{PA}(\mathbf{M}) = \mathbf{E}_{\mathbf{M}}\mathcal{L}(\overline{g}_n)$ for some Banach limit $\mathcal{L}$ on $I\!R^N$.

We study G_∞^{PA}, the game with strategy sets $(PA^i)_i$ and with payoff function $\tilde{g}^{PA}$, together with its set E^{PA} of equilibrium payoffs.

5 Trapdoor functions

Trapdoor functions, as introduced by Diffie and Hellman [DH76], are functions that are easy to compute, and for which it is computationally impossible to find an inverse. Only through the knowledge of certain **trapdoor information** can one easily compute the inverse.

Definition 2 *A trapdoor function* **TF** *is given by a finite set of messages X endowed with a probability π which is not a Dirac mass and by a family* $(\mathbf{MKD}, \mathbf{MKE}, \mathbf{ME}, \mathbf{MD})$ *of probabilistic polynomial Turing machines such that:*

- **MKD** *outputs some kd, called the* ***decoding key****.*
- *From input kd,* **MKE** *outputs ke, called the* ***deciphering key****.*
- *From $x \in X$ and ke,* **ME** *outputs y. We say that* **ME** ***enciphers*** *x into y.*
- *From y and kd,* **MD** *outputs x with probability one. (***MD** ***deciphers*** *the coded message y.)*

- *If* $\mathbf{M}$ *is a polynomial probabilistic Turing machine that outputs* $z \in X$ *from* y *and* ke*, then for every* $x_0 \in X$ *such that* $\pi(x_0) > 0$*:*

$$\lim_{t \to \infty} [P_t(z = x_0 | x = x_0) - P_t(z = x_0)] = 0$$

To this definition corresponds the following story:

Bob wants to send to Alice a message x, which is drawn in X according to π. A third person, Camilla, wants to know x. There exists no secure communication channel between Alice and Bob, so Camilla can listen to all of their conversation. The use of a trapdoor function allows Bob for sending private messages to Alice:

1. Alice picks a decoding key, kd.
2. Alice publicly announces ke, computed from kd.
3. Bob publicly announces y, computed from ke and x.
4. Alice computes x from y and kd.

The existence of polynomially bounded Turing machines **MKE**, **MKD**, **ME**, **MD** ensures that the computations that have to be done by Alice and by Bob are not too hard. Last condition means that Camilla has no easy way to extract any information on x from her information (ke and y).

Remark 5.1

There always exists an Turing machine $\mathbf{M}$ that computes x from ke and y with probability one. The principle is to try all the different possible combinations of x, ke, and kd until x is found. The important point is that $\mathbf{M}$ would take far more operations to compute x than what $\mathbf{MKD}, \mathbf{MKE}, \mathbf{ME}$ and **MD** need when the degree of complexity increases.

Remark 5.2

Assume there exists a trapdoor function for some finite set X endowed with probability π. Then, by applying the trapdoor function twice independently, we get the existence of a trapdoor function for the set of messages $X \times X$ endowed with probability $\pi \times \pi$.

Remark 5.3

If **TF** is a trapdoor function with the probability on X being π, and if π' is a probability on X that is absolutely continuous with respect to π, then **TF** is also a trapdoor function when the probability on X is π'. When referring to a trapdoor function, the probability π on X will generally remain unspecified.

Remark 5.4

The existence of a trapdoor function is a common assumption in cryptography, but has never been proved mathematically. Assume there exists a trapdoor function for a set of messages X endowed with some probability π, then the two last remarks show the existence of a trapdoor function for any finite set of messages endowed with any probability.

5.1 Example, the RSA cryptosystem

In 1977, Rivest, Shamir and Adleman [RSA78] proposed the following system for public cryptography:

The deciphering key kd consists of two large prime numbers p and q, together with a number $1 \leq e \leq (p-1)(q-1)$, e prime with $(p-1)(q-1)$. The enciphering key is the pair $n = (pq, e)$. To encipher some message $1 \leq x \leq pq$, one computes $y \equiv x^e \ [n]$. To decipher y, one first finds an inverse d of e modulo $\varphi(n) = (p-1)(q-1)$, where φ represents the Euler function. From elementary number theory, one has:

$$y^d \equiv x^{de} \equiv x^{k\varphi(n)+1} \equiv x \ [n]$$

To find x from y, n and e, no solution is known but factoring n, which is a hard problem. The degree of complexity is the number of digits of p and q.

6 Main result

Theorem 3 *If :(i)there exists a trapdoor function, and*

(ii) There exists $w \in F \cap CIR$ such that for every player i, $w^i > \underline{v}^i$,

then the closure of E^{PA} is $F \cap CIR$.

Proof: Any equilibrium payoff belongs to F. For $i \in I$, consider an Turing machine plays independently an action a_t^i distributed according to $\underline{s}^i$ at stage t. This guarantees $\underline{v}^i$ to player i in G_∞^{PA}. Hence $E^{PA} \subset CIR$.

To prove the converse inclusion, we consider a fixed $u \in F \cap CIR$, and $\eta > 0$. Assumption (ii) provides the existence of a rational barycenter v of elements in $g(A)$ ($v = \sum_{a \in A} p_a g(a)$ with p_a nonnegative rational and $\sum_{a \in A} p_a = 1$) such that $v^i > \underline{v}^i$ and $|v^i - u^i| < \eta$ for every i. We shall prove that v is an equilibrium payoff of G_∞^{PA}.

- First, we select a deterministic polynomial Turing machine that outputs periodically elements $a_t \in A$ according to the frequencies (p_a).
- For each i, we select a probabilistic polynomial Turing machine that draws $b_t^{-i} \in A^{-i}$ independently according to δ_{-i}. This was made possible by our (slightly non-standard) definition of a Probabilistic Turing Machine. Using the standard definition, one would only have to approximate δ_{-i} by distributions with diadic weights.
- We fix a trapdoor function $(\mathbf{MKD}, \mathbf{MKE}, \mathbf{ME}, \mathbf{MD})$ for the set of messages A endowed with any probability with full support.
- For each i, $a_0^i \in A^i$ represents a pure best response to δ_{-i}.
- For each i, $c(i)$ is an element of $N - \{i\}$, and $N(i)$ represents the set $N - \{i, c(i)\}$.

We shall prove that the following strategies form a Nash equilibrium of G_∞^{PA} inducing payoff v:

Main Path (MP): Play $\bar{a}_t$ at stage t. If i deviates from MP at stage t_0, go to $P(i)$

$P(i)$: Player $c(i)$ serves as a coordinator to punish i.

At every stage $t \geq t_0 + 1$, each player $j \in N(i)$ publicly sends a enciphering key, ke_{t+2}^j. The stage after, $c(i)$ draws a $N-1$-tuple of actions b_{t+2}^{-i} to be played at stage $t+2$ according to δ_{-i}. For all $j \in N(i)$, $c(i)$ codes b_{t+2}^{-i} with ke_{t+2}^i and announces the result publicly. Doing this, all players but i are informed after stage $t+1$ of the profile of actions b_{t+2}^{-i} to be played at stage $t+2$.

More precisely:

- At stage $t \geq t_0 + 1$, each player $j \in N(i)$ uses **MKD** to compute kd_{t+2}^j, then uses **MKE** to output ke_{t+2}^j from kd_{t+2}^j, and announces ke_{t+2}^j publicly.

- At stage $t \geq t_0 + 2$, $c(i)$ draws $b_{t+1}^{-i} \in A^{-i}$ according to δ_{-i}. Then, for all $j \in N(i)$, $c(i)$ uses **ME** to compute y_{t+1}^j from kd_{t+1}^j and $(b_{t+1}^{-i}, a_0^i) \in A$. The public message of $c(i)$ is the $N-2$-tuple of messages $(y_{t+1}^j)_{j \in N(i)}$.

- At stage $t \geq t_0 + 3$, each player $j \in N(i)$ uses **MD** to compute b_t^{-i} from kd_t^j and y_t^j, and plays b_t^j. Player $c(i)$ plays $b_t^{c(i)}$.

- At every stage $t \geq t_0 + 1$, player i sends no message and plays a_0.

Next table shows players $j \neq i$'s actions and messages at stage t:

	$i \in N(i)$	$c(i)$
Message	ke_{t+2}^j	$(y_{t+1}^j)_{j \in N(i)}$
Action	a_t^i	$a_t^{c(i)}$

The Turing machines defined above are polynomial since at each turn, they use a finite number of times one of a finite number of polynomial Turing machines.

The only information needed to play at stage t are the messages exchanged in the two previous turns, and whether some player deviated in the past. The State tapes are used to keep track of this information.

To prove that there these strategies form an equilibrium, it is enough to prove that for any polynomial Turing machine $\mathbf{M}^i \in PA^i$, the limit in the sense of $\mathcal{L}$ of the average payoffs for player i induced by $\mathbf{M}^i$ and by the strategies in $P(i)$ is less than $\underline{v}^i$.

To do this, we need to prove that in $P(i)$, player i cannot guess the actions of players in $N(i)$ from their messages. If there are three players, this is a direct consequence of the definition of a trapdoor function. With more that three players, we generalize the notion of a trapdoor function in the next subsection. Then, we show that a punished player cannot get more than his min max in correlated strategies against the above strategies, which completes the proof of Theorem 3.

6.1 k-trapdoor functions

Let k be a positive integer and $\mathbf{TF} = (\mathbf{MKD}, \mathbf{MKE}, \mathbf{ME}, \mathbf{MD})$ be a trapdoor function for some set of messages X endowed with probability π. $kd^1, \ldots, kd^k$ represent k independent outputs of **MKD**, and $ke^1, \ldots, ke^k$ are computed separately from $kd^1, \ldots, kd^k$ by **MKE**. From $x \in X$ drawn according to π and from $ke^1, \ldots, ke^k$, **ME** computes $y^1, \ldots, y^k$.

Definition 4 **TF** *is a* k***-trapdoor function*** *if for every probabilistic polynomial Turing machine that outputs* $z \in X$ *from* $ke^1, \ldots, ke^k, y^1, \ldots, y^k$ *and for every* $x_0 \in X$*:*

$$\lim_{t \to \infty} P_t(z = x_0 | x = x_0) - P_t(z = x_0) = 0$$

A k-trapdoor function is therefore a trapdoor function such that an outsider cannot get information on a message by observing it being coded k times.

Proposition 5 *Every trapdoor function is a k-trapdoor function for all k.*

We start to prove a lemma on trapdoor functions:

Lemma 6 *For every probabilistic polynomial Turing machine* $\mathbf{M}$ *that outputs z from ke^1 and y^1, for all $(x_0, x_1, x_2) \in X^3$ such that $\pi(x_1).\pi(x_2) > 0$:*

$$\lim_{t\to\infty} P_t(z = x_0 | x = x_1) - P_t(z = x_0 | x = x_2) = 0$$

Proof: From $\mathbf{M}$, we define the probabilistic polynomial Turing machine $\mathbf{M}'$ that inputs ke^1 and y^1 and:

- Computes z using $\mathbf{M}$.
- Outputs $z' = x_1$ if $z = x_0$, $z' = x_0$ otherwise.

Since **TF** is a trapdoor function:

$$\lim_{t\to\infty} P_t(z' = x_1 | x = x_1) - P_t(z' = x_1) = 0$$

which writes:

$$\lim_{t\to\infty} P_t(z = x_0 | x = x_1) - P_t(z = x_0) = 0$$

Applying the same procedure again with x_2 replacing x_1 gives:

$$\lim_{t\to\infty} P_t(z = x_0 | x = x_2) - P_t(z = x_0) = 0$$

Hence the result. ∎

Now, consider a probabilistic polynomial Turing machine $\mathbf{M}$ that outputs $z \in X$ from $ke^1, \ldots, ke^k$, $y^1, \ldots, y^k$. From $\mathbf{M}$, and for $(\bar{x}^1, \ldots, \bar{x}^k) \in X^k$, we define the probabilistic Turing machine $\bar{\mathbf{M}}$ (which depends on $\bar{x}^1, \ldots, \bar{x}^k$) that:

- Computes $\overline{kd}^1, \ldots, \overline{kd}^k$ using k times **MKD**
- Computes $\overline{ke}^1, \ldots, \overline{ke}^k$ from $\overline{kd}^1, \ldots, \overline{kd}^k$ using k times **MKE**
- Computes $\bar{y}^1, \ldots, \bar{y}^k$ from $\bar{x}^1, \ldots, \bar{x}^k$ and $\overline{ke}^1, \ldots, \overline{ke}^k$ using k times **ME**
- Outputs $\bar{z} = \bar{z}(\bar{x}^1, \ldots, \bar{x}^k)$ from $\bar{y}^1, \ldots, \bar{y}^k$, $\overline{ke}^1, \ldots, \overline{ke}^k$ using **M**.

Given $x_0 \in X$ such that $\pi(x_0) > 0$, we denote $f_t(\bar{x}^1, \ldots \bar{x}^k) = P_t(\bar{z} = x_0)$.

Lemma 7 *Assume* $\pi(\bar{x}^1) \ldots \pi(\bar{x}^k) > 0$ *and* $\pi(\bar{x}'^1) \ldots \pi(\bar{x}'^k) > 0$, *then:*

$$\lim_{t \to \infty} f_t(\bar{x}^1, \ldots \bar{x}^k) - f_t(\bar{x}'^1, \ldots \bar{x}'^k) = 0$$

Proof: It is enough to prove that for any $1 \leq p \leq k$, if $\pi(\bar{x}^1) \ldots \pi(\bar{x}^k).\pi(\bar{x}'^p) > 0$:

$$\lim_{t \to \infty} f_t(\bar{x}^1, \ldots \bar{x}^k) - f_t(\bar{x}^1, \ldots, \bar{x}^{p-1}, \bar{x}'^p, \bar{x}^{p+1}, \ldots, \bar{x}^k) = 0$$

To keep notations simple we prove it for $p = 1$. From $\bar{x}^2, \ldots, \bar{x}^k$, we define $\bar{\mathbf{M}}'$ (depending on $\bar{x}^2, \ldots, \bar{x}^k$) as the polynomial Turing machine that inputs ke^1 and y^1, and:

- Computes $\overline{kd}^2, \ldots, \overline{kd}^k$ using $k-1$ times **MKD**
- Computes $\overline{ke}^2, \ldots, \overline{ke}^k$ from $\overline{kd}^2, \ldots, \overline{kd}^k$ using $k-1$ times **MKE**
- Computes $\bar{y}^2, \ldots, \bar{y}^k$ from $\bar{x}^2, \ldots, \bar{x}^k$ and $\overline{ke}^2, \ldots, \overline{ke}^k$ using $k-1$ times **ME**
- Outputs $\bar{z}' = \bar{z}'(\bar{x}^2, \ldots, \bar{x}^k)$ from $y^1, \bar{y}^2 \ldots, \bar{y}^k$, $ke^1, \ldots, \overline{ke}^k$ using M.

Since **TF** is a trapdoor function, from Lemma 6 we have:

$$\lim_{t \to \infty} P_t(\bar{z}' = x_0 | x = \bar{x}^1) - P_t(\bar{z}' = x_0 | x = \bar{x}'^k) = 0$$

Observing that $P_t(\bar{z}' = x_0 | x = \bar{x}^1) = f_t(\bar{x}^1, \ldots \bar{x}^k)$ and $P_t(\bar{z}' = x_0 | x = \bar{x}'^1) = f_t(\bar{x}'^1, \bar{x}^2, \ldots, \bar{x}^k)$ then gives the result. ∎

Proof of Proposition 5:

Consider x_0 and x_1 such that $\pi(x_0).\pi(x_1) > 0$, and apply Lemma 7 to $\bar{x}^1 = \bar{x}^2 \ldots = \bar{x}^k = x_0$, $\bar{x}'^1 = \bar{x}'^2 \ldots = \bar{x}'^k = x_1$. One gets:

$$\lim_{t \to \infty} P_t(z = x_0 | x = x_0) - P_t(z = x_0 | x = x_1) = 0$$

As we have

$$P_t(z = x_0) = \sum_{x_1 \in X} \pi(x_1) P_t(z = x_0 | x = x_1)$$

this implies

$$\lim_{t \to \infty} P_t(z = x_0 | x = x_0) - P_t(z = x_0) = 0$$

∎

Corollary 8 *For any set B, for any probabilistic polynomial Turing machine $\mathbf{M}_B$ and that outputs $b \in B$ from $y^1, \ldots, y^k$ and $ke^1, \ldots, ke^k$, for every $x_0 \in X$ and b_0 in B:*

$$\lim_{t \to \infty} P_t(b = b_0 | x = x_0) - P_t(b = b_0) = 0$$

Proof: The proof is identical to the proof of Lemma 6, except that the trapdoor function is replaced by a k-trapdoor function. ∎

Corollary 9 *For every finite set B, for every mapping $h : X \times B \to \mathbb{R}$ and for every probabilistic polynomial Turing machine that outputs $b \in B$ from $ke^1, \ldots, ke^k$, $y^1, \ldots, y^k$:*

$$\lim_{t \to \infty} \mathbf{E}_t h(x_0, b) \leq \max_{b_0 \in B} \mathbf{E}_t h(x_0, b_0)$$

where $\mathbf{E}_t$ denotes the expectation operator under P_t.

Proof: Fix $\varepsilon > 0$, there exists t_0 such that for all $t \geq t_0$, $b_0 \in B$ and $x_0 \in X$:

$$|P_t(b = b_0 | x = x_0) - P_t(b = b_0)| \leq \varepsilon$$

Let also C be a bound of h on $X \times B$. Then

$$\begin{aligned}
\mathbf{E}_t h(x,b) &= \sum_{x_0} \sum_{b_0} P_t(x = x_0) P_t(b = b_0 | x = x_0) h(x_0, b_0) \\
&\leq \sum_{x_0} \sum_{b_0} P_t(x = x_0) P_t(b = b_0) h(x_0, b_0) + \varepsilon C \\
&\leq \sum_{b_0} P_t(b = b_0) \max_{b_1} \sum_{x_0} P_t(x = x_0) h(x_0, b_1) + \varepsilon C \\
&\leq \max_{b_1} \mathbf{E}_t h(x, b_1) + \varepsilon C
\end{aligned}$$

which proves the corollary. ∎

With $B = A^i$, $X = A^{-i}$, and $h = g^i$, Corollary 9 implies that a player i who is limited to strategies in PA^i cannot get more than $\underline{v}^i$ against the strategies defined in $P(i)$. This completes the proof of Theorem 3.

7 Subgame perfect equilibria

In order to extend our result to subgame perfect equilibria, one first needs to provide a definition of a subgame equilibrium for games played by polynomial Turing machines. The standard definition of subgame perfection asks that strategies form an equilibrium of the remaining game whatever the past history. In our model, the private history tapes are used to keep track of historical information for each agent. (These tapes may be different among the agents so strictly speaking, no common knowledge sets may exist.) Assume h_t^i represents the stream of symbols on the private history of Turing machine $\mathbf{M}^i$ at the beginning of stage t. Given t and a family of private histories $h_t = (h_t^i)_i$ at stage t, any family of polynomial Turing machines

$\mathbf{M}'=(\mathbf{M}'^i)_i$ induces a probability on future actions, and thus the expected limit of continuation payoffs $\tilde{g}^{PA}_{h_t}(\mathbf{M}')$ is well defined as in Section 4.2. We call $\mathbf{M}$ a subgame perfect equilibrium of G^{PA}_∞ when $\mathbf{M}$ is a Nash equilibrium of the game with strategy spaces $(PA^i)_i$ and payoff function $\tilde{g}^{PA}_{h_t}$ for every t and every h_t. Let E'^{PA} be the set of payoffs in G^{PA}_∞ induced by subgame perfect equilibria. Note that our definition of subgame perfection is somehow stronger than the usual one since the private histories of different players need not be consistent one with the other.

Theorem 10 *If :(i)there exists a trapdoor function, and*
(ii) There exists $w \in F \cap CIR$ such that for every player i, $w^i > \underline{v}^i$,
then the closure of E'^{PA} is $F \cap CIR$.

We prove this result by modifying the strategies defined in Section 6, in such a way that they define subgame perfect equilibria.

First, we must assume that punishment phases are finite but of longer and longer lengths. If a player deviates at stage τ, his punishment phase lasts until period 2τ. This ensures that no deviation can be profitable in the long run, but still each punishment phase has a finite duration so that no punisher has incentives to deviate from the punishment phase.

Second, since private history tapes may not be consistent with each other, we must decide of a rule so that players can agree to follow the same equilibrium path. Otherwise players could punish one another forever.

- In case there are at least three player: at each stage t players announces who deviated last and at what stage according to his private history. A majority rule is then applied to these messages. If a majority says some player i deviated at stage τ and if $t \leq 2\tau$, then i is punished until stage 2τ. Otherwise the Main Path is played.

- In case there are two players: players make public announcements as in the previous case.

- If no player says a punishment should take place the Main Path goes on.
- If one player says a punishment phase should take place until stage 2τ and the other says no punishment should take place, then each player plays a min max strategy against the other until stage 2τ.
- If one player says a punishment phase should take place until stage $2\tau^1$ while the other says a punishment should take place until stage $2\tau^2$, then each player plays a min max strategy until $\max(2\tau^1, 2\tau^2)$.

With at least three players, no player can profitably cheat by making a fake announcement since a majority rule applies. With two players, the punishments are designed so that no player can avoid being punished if he deviated in the past nor can he gain by pretending the other should be punished since punishments are reciprocal.

8 Concluding remarks

We have explored some consequences of modern cryptography to a model of infinitely repeated games without discounting. Our main result states that if public communication is allowed and if we can apply the assumptions of modern cryptography, then the equilibrium payoffs of the game are the correlated equilibrium payoffs of the usual repeated game.

First, note that no equivalent result could hold in a finitely repeated game or in an infinitely repeated game with discounting. Actually, if we fix strategies for players $j \neq i$ and a finite horizon n, there exists a polynomial Turing machine for player i that allows him to defend his min max in mixed strategies during the n first stages of the game. The reason for this is that the limitation on the computation power is only effective when n tends to ∞. Therefore, infinitely repeated games with no discounting seem the

appropriate framework to model players using modern cryptography.

We assumed that all players have access to a public communication device. A main extension would be to prove a similar result in which all the correlation would go through the actions played in the game. Or equivalently, one can assume that the sets of messages are finite instead of countable here. The most natural way to prove such a result is to divide punishment phases into communication phases and action phases. Then, the major difficulty is to keep the size of the former small compared to the size of the latter. For this purpose, pseudo-random generators, which allow to construct a long sequence of numbers that look random from a relatively small seed seem to be the adequate tool.

Our main result shows that coordination between any subgroup of the players may arise even when the only communication available is public communication. This strengthens the idea that correlated equilibria may arise in general setups, even without private signals or when no private communication channels are available.

References

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithm.* Addison-Wesley, Reading, MA, 1974.

[AS94] R.J. Aumann and L.S. Shapley. Long-term competition—A game theoretic analysis. In N. Megiddo, editor, *Essays on Game Theory in Honor of Michael Maschler*, pages 1–15. Springer-Verlag, New-York, 1994.

[Aum81] R.J. Aumann. *Survey of Repeated Games*, pages 11–42. Wissenschaftsverlag, Bibliographisches Institut, Mannheim, Wien, Zurich, 1981.

[Axe84] R. Axelrod. *The evolution of cooperation.* Basic Books, New-York, 1984.

[Bin87] K.G. Binmore. Modeling rational players – part I. *Economics and Philosophy*, 3:179–214, 1987.

[Bin88] K.G. Binmore. Modeling rational players – part II. *Economics and Philosophy*, 4:9–55, 1988.

[BP93] E. Ben Porath. Repeated games with finite automata. *Journal of Economic Theory*, 59:17–32, 1993.

[BS92] K.G. Binmore and L. Samuelson. Evolutionary stability in repeated games. *Journal of Economic Theory*, 57:278–305, 1992.

[DH76] W. Diffie and E. Hellman. New directions in cryptography. In *IEEE transactions on information theory*, volume IT-22, pages 644–654, 1976.

[For90] F. Forges. Universal mechanisms. *Econometrica*, 58:1341–1364, 1990.

[LP81] H. R. Lewis and C. H. Papadimitriou. *Elements of the theory of computation.* Prentice-Hall, 1981.

[Meg86] N. Megiddo. Remarks on bounded rationality. I.B.M research paper RJ 5270, 1986.

[Mer80] J.-F. Mertens. A note on the characteristic function of supergames. *International Journal of Game Theory*, 9:189–190, 1980.

[Ney85] A. Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoner's dimema. *Economic Letters*, 19:227–229, 1985.

[Ney98] A. Neyman. Finitely repeated games with finite automata. *Mathematics of Operations Research*, 23:513–552, 1998.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Rub86] A. Rubinstein. Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory*, 39:83–86, 1986.

[Rub94] A. Rubinstein. Equilibrium in supergames. In N. Megiddo, editor, *Essays on Game Theory in Honor or Michael Maschler*, pages 17–27, Berlin, 1994. Springer-Verlag.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 ; 623–656, 1948.

[UV97] A. Urbano and J. E. Vila. Pre-play communication and coordination in two-player games. WP-AD 97-26, Instituto Valenciano de Investigaciones Economicas, 1997.

[UV98a] A. Urbano and J. E. Vila. Unmediated communication in repeated games with imperfect monitoring. WP-AD, Instituto Valenciano de Investigaciones Economicas, 1998.

[UV98b] A. Urbano and J. E. Vila. Unmediated talk under incomplete information. WP-AD, Instituto Valenciano de Investigaciones Economicas, 1998.