CORE DISCUSSION PAPER
9835

REPEATED GAMES PLAYED BY CRYPTOGRAPHICALLY SOPHISTICATED PLAYERS[1]

Olivier Gossner[2]

June, 1998

Abstract

We explore the consequences of the assumptions used in modern cryptography when applied to repeated games with public communication. Technically speaking, we model agents by polynomial Turing machines and assume the existence of a trapdoor function. Under these conditions, we prove a Folk Theorem in which the min max level of players has to be taken in correlated strategies instead of mixed strategies..

KEYWORDS: repeated games, bounded rationality, correlation, Turing machines, cryptography

J.E.L. CLASSIFICATION: C72

---

1 The author is very grateful to Nathan Linial, Jean-François Mertens and Jacques Stern for their advices and to Gaetano Bloise for his helpful comments..

2 CORE, 34 Voie du roman pays, Louvain-la-Neuve, B-1348. gossner@core.ucl.ac.be

This text presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsability is assumed by the author.

1 Introduction

Bounded rationality models study the implications of the limitedness of the rationality of the agents on the outcomes of games. Modern cryptography, which finds many applications into everyday life, relies on the assumption that agent's computational powers are limited, and represents them by polynomial Turing machines. We explore the consequences of such an assumption when applied to repeated games with public communication.

Cryptography deals with security of communication. The central problem is the one of two agents A and B who 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 cannot understand them. To send the information x to B, A uses a key k_e to encode x into $y = k_e(x)$ and sends message y to B. After receiving y, B uses a key k_d 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 the knowledge of y without possessing information on k_d.

We can distinguish two forms of cryptography: classical cryptography, which dates back at least to the times of Julius Cæsar, and modern cryptography, which was initiated in 1976 by Diffie and Hellman [DH76].

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 room 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 to such classical cryptosystems is that the length (or more precisely the entropy) of x cannot exceed the length of the keys, otherwise condition [1] and [2] cannot hold together. This limitation on the size of x can be easily seen in our example.

In modern cryptography A and B share no secret information before they communicate. In typical modern cryptosystems, B chooses k_d, computes k_e from k_d, then sends k_e to A (or disclaims it publicly). Messages are then sent from A to B using the pair of keys (k_e, k_d). Usually, there is 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. Here intervenes 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. For instance, some 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 less than a minute each 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 inverse without knowing the trapdoor information k_d.

Turing machines is the model of computation commonly used by computer scientists. According to Church's thesis, any algorithm known can be represented by a Turing machine. References of games played by Turing machines date back at least to Rabin [Rab53], but Binmore ([Bin87], [Bin88]) was the first to propose Turing machines as a model of bounded rationality. Some implications of Turing machines into repeated games were then studied by Anderlini and Sabourian [AS95]. We defend here the idea that not only should player's strategies be implementable by some algorithms, but also these algorithms should not take a too long time for their computations. Consider a Turing that inputs the description of a task to perform together with some degree of complexity. Such a machine is called polynomial when the time 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 represented by polynomial Turing machines..

Problems that cannot be solved by Turing machines are called undecidable (no algorithm can solve them), whereas those that cannot be solved by polynomial Turing machines are called intractable (they require too much computation in practice). By limiting player's strategies to polynomial Turing machines, we simply mean that they are not capable of performing operations that would take one century to compute by a big computer. On the other hand, we believe each of them can use a common home computer to help him take his decisions.

Our main result is a "Folk Theorem" in which the usual min max level in mixed strategies is replaced by the min max in correlated strategies. The key is that any group of players can generate private information through public messages using a modern cryptosystem. Hence deviators can be punished in correlated strategies. We shall exhibit particular strategies in which when a

player deviates, another player is designed to coordinate the punishment. At each turn, the coordinator draws a correlated action of the punishers against the punished player. A modern cryptosystem is then used to transmit this profile of punishing actions to each of the punishers. It remains to prove that the punished player cannot get any information on this profile of actions from the communication between the coordinator and each of the other punishers. This is done by extending the notion of a trapdoor functions (which allows some agent A to send a secret message to another player B) into k-trapdoor functions (which allow A to send a secret message separately to any group of k other players). Proving that any trapdoor function is also a k-trapdoor function is the technical lemma on which the proof of our main theorem relies.

To our knowledge, the only application so far of modern cryptography to game theory was done by Urbano and Vila [UV97]. They exhibit protocols through which two players can correlate through noiseless communication (this problem is often referred to as "telephone coin flipping" or "mental poker"). As opposed to them, the strategies we exhibit do not rely on a particular protocol (the development of which has led to a huge literature in cryptography), but on the theoretical assumption of the existence of a trapdoor function. Our aim is then to explore some consequences of modern cryptography when applied to the classical game theoretic model of repeated games.

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

2 Turing machines

A Turing machine $\mathbf{M}$ 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. $\mathbf{M}$ 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 for computation one may think of can be implemented by a Turing machine. This result constitutes Church's thesis.

Note that it is yet not true that any function can be computed by a Turing machine. Rabin [Rab53] shows that some win-lose sequential games

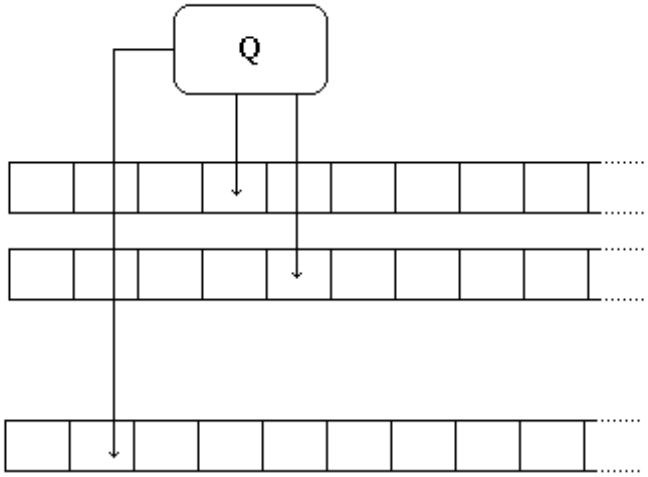

Figure 1: head and tapes of a Turing machine

may have no equilibria if we require player's strategies to be implemented by Turing machines. Negative results in the same vein are obtained by Anderlini [And90].

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].

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

2.2 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.

2.3 Polynomial time 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 non deterministic, we require this inequality to hold for each* $\tilde{s}$ *with probability one.*

2.4 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.

3 Turing machines playing repeated games

3.1 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.

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:

$$\begin{aligned} \underline{v}^i &= \min_{\beta \in \Delta(A^{-i})} \max_{s^i \in S^i} \mathbf{E}_\beta \, g^i(a^{-i}, s^i) && (1) \\ &= \max_{s^i \in S^i} \min_{s^{-i} \in S^{-i}} g^i(s^{-i}, s^i) && (2) \end{aligned}$$

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.

$CIR = \{x \in I\!R^N, \forall i \; x^i \geq \overline{v}^i\}$ 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 $IR \subset CIR$.

3.2 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.

4 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 **trap-door 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 deterministic 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** ensure that the computations that have to be done by Alice and by Bob are not too hard. Last condition of the definition ensures that Camilla has no easy way to guess x from her information (ke and y).

Remark 4.1

There always exists an Turing machine **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 **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 4.2

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

Remark 4.3

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 4.4

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.

4.1 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 random number $1 \leq e \leq pq$. 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.

5 Main result

Theorem 1 *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}.
- 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:

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 kd_{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}^i, then uses **MKE** to output ke_{t+2}^i from kd_{t+2}^i, and announces ke_{t+2}^i 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^j_{t+2}	$(y^j_{t+1})_{j \in N(i)}$
Action	a^i_t	$a^{c(i)}_t$

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 defintion 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 that his min max in correlated strategies against the above strategies, which completes the proof of Theorem 1.

5.1 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 3 **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 2 *Every trapdoor function is a k-trapdoor function for all k.*

We start to prove a lemma on trapdoor functions:

Lemma 3 *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 4 *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 3 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 2:
Consider x_0 and x_1 such that $\pi(x_0).\pi(x_1) > 0$, and apply Lemma 4 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 5 *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 3, except that the trapdoor function is replaced by a k-trapdoor function. ■

Corollary 6 *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 6 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 1.

6 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 $\tilde{g}_{h_t}^{PA}(\mathbf{M}')$ is well defined as in Section 3.2. 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 $\tilde{g}_{h_t}^{PA}$ 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 7 *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 5, in such a way that they define subgame perfect equilibria.

First, as is the proof of the perfect "Folk Theorem" without discounting due to Aumann and Shapley [AS94] and Rubinstein [Rub77], we must assume that punishment phases are finite but of longer and longer lengths. So, if a player deviates at stage τ, his punishment phase lasts until period 2τ. 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 announcments 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 announcment 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.

7 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 tends to strengthen 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.

[And90] L. Anderlini. Some notes on Church's thesis and the theory of games. *Theory and Decision*, 29:19–52, 1990.

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

[AS95] L. Anderlini and H. Sabourian. Cooperation and effective computability. *Econometrica*, 63:1337–1369, 1995.

[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.

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

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

[Rab53] O. Rabin. Effective computability of winning strategies. In *Contributions to the theory of games*, volume 3 of *Annals of Mathematical Studies*, pages 147–157. Princeton: Princeton University Press, 1953.

[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.

[Rub77] A. Rubinstein. Equilibrium in supergames. Center for Research in Mathematical Economics and Game Theory, Research Memorandum 25, 1977.

[UV97] A. Urbano and J. E. Vila. Pre-play communication and coordination in two-player games. handout, 1997.