The Truth Behind the Myth of the Folk Theorem

Joseph Y. Halpern Rafael Pass Lior Seeman
Computer Science Dept.
Cornell University
Ithaca, NY
E-mail: halpern|rafael|lseeman@cs.cornell.edu

Abstract

We study the problem of computing an ϵ-Nash equilibrium in repeated games. Earlier work by Borgs et al. [2010] suggests that this problem is intractable. We show that if we make a slight change to their model—modeling the players as polynomial-time Turing machines that maintain state (rather than stateless polynomial-time Turing machines)—and make some standard cryptographic hardness assumptions (the existence of public-key encryption), the problem can actually be solved in polynomial time.

1 Introduction

The complexity of finding a Nash equilibrium (NE) is a fundamental question at the interface of game theory and computer science. A celebrated sequence of results showed that the complexity of finding a NE in a normal-form game is PPAD-complete [Chen and Deng 2006; Daskalakis, Goldberg, and Papadimitriou 2006], even for 2-player games. Less restrictive concepts, such as ϵ-NE for an inverse-polynomial ϵ, are just as hard [Chen, Deng, and Teng 2006]. This suggests that these problems are computationally intractable.

There was some hope that the situation would be better in infinitely-repeated games. The *Folk Theorem* (see [Osborne and Rubinstein 1994] for a review) informally states that in an infinitely-repeated game G, for any payoff profile that is *individually rational*, in that all players get more than[1] their minimax payoff (the highest payoff that a player can guarantee himself, no matter what the other players do) and is the outcome of some correlated strategy in G, there is a Nash equilibrium of G with this payoff profile. With such a large set of equilibria, the hope was that finding one would be less difficult. Indeed, Littman and Stone [2005] showed that these ideas can be used to design an algorithm for finding a NE in a two-player repeated game.

Borgs et al. [2010] (BC+ from now on) proved some results suggesting that, for more than two players, even in infinitely-repeated games it would be difficult to find a NE. Specifically, they showed that, under certain assumptions, the problem of finding a NE (or even an ϵ-NE for an inverse-polynomial ϵ) in an infinitely repeated game with three or more players where there is a discount factor bounded away from 1 by an inverse polynomial is also PPAD-hard. They prove this by showing that, given an arbitrary normal-form game G with $c \geq 2$ players, there is a game G' with $c+1$ players such that finding an $\epsilon/8c$-NE for the repeated game based on G' is equivalent to finding an ϵ-NE for G.

While their proof is indeed correct, in this paper, we challenge their conclusion. Not surprisingly, we do this by changing their assumptions in what we argue are natural ways. Like BC+, we assume

[1]For our results, since we consider ϵ-NE, we can replace "more than" by "at least".

that players are resource bounded.[2] Formally, we view players as probabilistic[3] polynomial-time Turing machines (PPT TMs). We differ from BC+ in two key respects. First, BC+ implicitly assume that players have no memory: they cannot remember computation from earlier rounds. By way of contrast, we allow players to have a bounded (polynomial) amount of memory. This allows players to remember the results of a few coin tosses from earlier rounds, and means that we can use some cryptography (making some standard cryptographic assumptions) to try to coordinate the players. We stress that this coordination happens in the process of the game play, not through communication. That is, there are no side channels; the only form of "communication" is by making moves in the game. We call such TMs *stateful*, and the BC+ TMs *stateless*. Second, since we restrict to (probabilistic) polynomial-time players, we restrict the deviations that can be made in equilibrium to those that can be computed by such players; BC+ allow arbitrary deviations. Without this extra restriction, there is no real difference between stateful TMs and stateless TMs in our setting (since a player with unbounded computational power can recreate the necessary state). With these assumptions (and the remaining assumptions of the BC+ model), we show that in fact an ϵ-NE in an infinitely-repeated game can be found in polynomial time.

Roughly speaking, the ϵ-NE can be described as proceeding in three stages. In the first stage, the players play a sequence of predefined actions repeatedly. If some player deviates from the sequence, the second stage begins, in which the other players use their actions to secretly exchange a random seed, through the use of public-key encryption. In the third stage, the players use a correlated minimax strategy to punish the deviator forever. To achieve this correlation, the players use the secret random seed as the seed of a pseudorandom function, and use the outputs of the pseudorandom function as the source of randomness for the correlated strategy. Since the existence of public-key encryption implies the existence of pseudorandom functions, the only cryptographic assumption needed is the existence of public-key encryptions—one of the most basic cryptographic hardness assumptions.

1.1 Related work

The idea of considering resource-bounded agents has a long history in game theory. It is known, for example, that cooperation is a NE of finitely-repeated prisoner's dilemma with resource-bounded players (see, e.g., [Neyman 1985; Rubinstein 1986; Papadimitriou and Yannakakis 1994]). The idea of using the structure of the game as a means of correlation is used by Lehrer [1991] to show an equivalence between NE and correlated equilbrium in certain repeated games with nonstandard information structures. The use of cryptography in game theory goes back to Urbano and Vila [2002, 2004], who also used it to do coordination between players. More recently, it has been used by, for example, Dodis, Halevi, and Rabin [2000].

The application of cryptography perhaps most closely related to ours is by Gossner [1998], who uses cryptographic techniques to show how any payoff profile that is above the players' *correlated* minimax value can be achieved in a NE of a repeated game with public communication played by computationally bounded players. In [Gossner 2000], a strategy similar to the one that we use is used to prove that, even without communication, the same result holds. Gossner's results apply only to infinitely-repeated games with 3 players and no discounting; he claims that his results do not hold for games with discounting. Gossner does not discuss the complexity of finding a strategy of the type that he shows exists.

[2] Although BC+ do not discuss modeling players in this way, the problem they show is NP-Hard is to find a polynomial-time TM profile that implements an equilibrium. There is an obvious exponential-time TM profile that implements an equilibrium: each TM in the profile just computes the single-shot NE and plays its part repeatedly.

[3] BC+ describe their TMs as deterministic, but allow them to output a mixed strategy. As they point out, there is no difference between this formulation and a probabilistic TM that outputs a specific action; their results hold for such probabilistic TMs as well.

Recently, Andersen and Conitzer [2013] described an algorithm for finding NE in repeated games with more than two players with high probability in *uniform games*. However, this algorithm is not guaranteed to work for all games, and uses the limit of means as its payoff criterion, and not discounting.

2 Preliminaries

2.1 Infinitely repeated games

We define a game G as a triple $([c], A, \vec{u})$, where $[c] = \{1, \ldots, c\}$ is the set of players, A_i is the set of possible actions for player i, $A = A_1 \times \ldots \times A_c$ is the set of action profiles, and $\vec{u} : A \to \mathbb{R}^c$ is the utility function ($\vec{u}_i(\vec{a})$ is the utility of player i). A (mixed) *strategy* σ_i for player i is a probability distribution over A_i, that is, an element of $\Delta(A_i)$ (where, as usual, we denote by $\Delta(X)$ the set of probability distributions over the set X). We use the standard notation $\vec{x}_{-i}$ to denote vector $\vec{x}$ with its ith element removed, and $(x', \vec{x}_{-i})$ to denote $\vec{x}$ with its ith element replaced by x'.

Definition 2.1. *(Nash Equilibrium) $\sigma = (\sigma_1, ..., \sigma_c)$ is an ϵ-NE of G if, for all players $i \in [c]$ and all actions $a'_i \in A_i$,*

$$E_{\sigma_{-i}}[u_i(a'_i, \vec{a}_{-i})] \leq E_{\sigma}[u_i(\vec{a})] + \epsilon.$$

A correlated strategy of a game G is an element $\sigma \in \Delta(A)$. It is a *correlated equilibrium* if, for all players i, they have no temptation to play a different action, given that the action profile was chosen according to σ. That is, for all players i for all $a_i \in A_i$ such that $\sigma_i(a_i) > 0$, $E_{\sigma|a_i} u_i(a_i, \vec{a}_{-i}) \geq E_{\sigma|a_i} u_i(a'_i, \vec{a}_{-i})$.

Player i's minimax value in a game G is the highest payoff i can guarantee himself if the other players are trying to push his payoff as low as possible. We call the strategy i plays in this case a minimax strategy for i; the strategy that the other players use is i's (correlated) punishment strategy. (Of course, there could be more than one minimax strategy or punishment strategy for player i.) Note that a correlated punishment strategy can be computed using linear programming.

Definition 2.2. *Given a game $G = ([c], A, \vec{u})$, the strategies $\vec{\sigma}_{-i} \in \Delta(A_{-i})$ that minimize $\max_{\sigma' \in \Delta(A_i)} E_{(\sigma', \vec{\sigma}_{-i})}[u_i(\vec{a})]$ are the* punishment strategies *against player i in G. If $\vec{\sigma}_{-i}$ is a punishment strategy against player i, then $mm_i(G) = \max_{a \in A_i} E_{\vec{\sigma}_{-i}}[u_i(a, a_{-i})]$ is player i's* minimax value *in G*

Simplifying assumption: We normalize all payoffs so that each player's minimax value is 0. Since, in an equilibrium, all players get at least their minimax value, this guarantees that all players get at least 0 in a NE. We also assume that each player has at least two actions in G. (This allows us to use the actions played in the infinitely-repeated game based on G to encode bit strings.) This assumption is without loss of generality—we can essentially ignore players for whom it does not hold.

Given a normal-form game G, we define the repeated game $G^t(\delta)$ as the game in which G is played repeatedly t times (in this context, G is called the *stage game*) and $1 - \delta$ is the discount factor (see below). Let $G^\infty(\delta)$ be the game where G is played infinitely many times. An infinite history h in this game is an infinite sequence $\langle \vec{a}^0, \vec{a}^1, \ldots \rangle$ of actions profiles. Intuitively, we can think of $\vec{a}^t$ as the action profile played in the t^{th} stage game. We often omit the δ in $G^\infty(\delta)$ if it is not relevant to the discussion. Like BC+, we assume that G^∞ is *fully observable*, in the sense that, after each stage game, the players observe exactly what actions the other players played.

Since we consider computationally-bounded players, we take a player's strategy in G^∞ to be a (possibly probabilistic) Turing machine (TM), which outputs at each round an action to be played, based on its internal memory and the history of play so far. (The TMs considered in BC+ did not

have internal memory.) We consider only TMs that at round t use polynomial in nt many steps to compute the next action, where n is the maximum number of actions a player has in G. Thus, n is a measure of the size of G.[4] Denote by M_i the TM used by player i, and let $\vec{M} = (M_1, \ldots, M_c)$.

Note that a profile $\vec{M}$ induces a distribution $\rho_{\vec{M}}$ on infinite histories of play. Let $\rho^t_{\vec{M}}$ denote the induced distribution on H^t, the set of histories of length t. (If $t = 0$, we take H^0 to consist of the unique history of length 0, namely $\langle\rangle$.) Player i's utility if $\vec{M}$ is played, denoted $p_i(\vec{M})$, is defined as follows:

$$p_i(\vec{M}) = \delta \sum_{t=0}^{\infty} (1-\delta)^t \sum_{h \in H^t, \vec{a} \in A} \rho^{t+1}_{\vec{M}}(h \cdot \vec{a})[u_i(\vec{a})].$$

Thus, the discount factor is $1 - \delta$. Note that the initial δ is a normalization factor. It guarantees that if $u_i(\vec{a}) \in [b_1, b_2]$ for all joint actions $\vec{a}$ in G, then i's utility is in $[b_1, b_2]$, no matter which TM profile $\vec{M}$ is played.

We are now ready to define the notion of equilibrium we use. Intuitively, as we model players as polynomial-time TMs, we consider a profile of TMs an equilibrium in a game if there is no player and no other polynomial-time TM that gives that player a higher expected payoff (or up to an ϵ for an ϵ-NE).

Since we consider (probabilistic) TMs that run in polynomial time in the size of the game, we cannot consider a single game. For any fixed game, running in polynomial time in the size of the game is meaningless. Instead, we need to consider a sequence of games. This leads to the following definition.

Definition 2.3. *An infinite sequence of strategy profiles $\vec{M}^1, \vec{M}^2, \ldots$, where $\vec{M}^k = (M^k_1, ..., M^k_c)$, is an ϵ-NE of an infinite sequence of repeated games $G^\infty_1, G^\infty_2, \ldots$ where the size of G_k is k, if, for all players $i \in [c]$ and all infinite sequences of polynomial-time TMs $M^1, M^2, \ldots$ (polynomial in n and t, as discussed above), there exists k_0 such that, for all $k \geq k_0$,*

$$p^k_i(M^k, \vec{M}^k_{-i}) \leq p^k_i(\vec{M}^k) + \epsilon(k),$$

where p^k_i is the payoff of player i in game G^∞_k.

We note that the equilibrium definition we use considers only deviations that can be implemented by polynomial-time TMs. This is different from both the usual definition of NE and from the definition used by BC+, who allow arbitrary deviations. The need to define polynomial-time deviation is the reason for considering sequences of games instead of a single game. There are other reasonable ways of capturing polynomial-time adversaries. As will be seen from our proof, our approach is quite robust, so our results should hold for any reasonable definition.

2.2 Cryptographic definitions

For a probabilistic algorithm A and an infinite bit string r, $A(x; r)$ denotes the output of A running on input x with randomness r; $A(x)$ denotes the distribution on outputs of A induced by considering $A(x; r)$, where r is chosen uniformly at random. A function $\epsilon : \mathbb{N} \to [0, 1]$ is *negligible* if, for every constant $c \in \mathbb{N}$, $\epsilon(k) < k^{-c}$ for sufficiently large k.

In this section, when we mention a PPT algorithm, we mean a *non-uniform* PPT algorithm and, specifically, an algorithm that, in addition to its regular input, gets for every input length an additional input (of polynomial length) that is viewed as *advice* . It is common to assume that the cryptographic building blocks we define next and use in our constructions are secure against non-uniform PPT algorithms.

[4] When we talk about polynomial-time algorithms, we mean polynomial in n. We could use other measures of the size of G, such as the total number of actions. Since all reasonable choices of size are polynomially related, the choice does not affect our results.

2.2.1 Computational Indistinguishability

Definition 2.4. *A* probability ensemble *is a sequence $X = \{X_n\}_{n\in\mathbb{N}}$ of random variables indexed by $\mathbb{N}$. (Typically, in an ensemble $X = \{X_n\}_{n\in\mathbb{N}}$, the domain of X_n consists of strings of length n.)*

We now recall the definition of computational indistinguishability [Goldwasser and Micali 1984].

Definition 2.5. *Two probability ensembles $\{X_n\}_{n\in\mathbb{N}}, \{Y_n\}_{n\in\mathbb{N}}$ are* computationally indistinguishable *if, for all PPT algorithms D, there exists a negligible function ϵ such that, for all $n \in \mathbb{N}$,*

$$|\Pr[D(1^n, X_n) = 1] - \Pr[D(1^n, Y_n) = 1]| \leq \epsilon(n).$$

To explain the Pr *in the last line, recall that X_n and Y_n are random variables. Although we write $D(1^n, X_n)$, D is a randomized algorithm, so what $D(1^n, X_n)$ returns depends on the outcome of random coin tosses. To be a little more formal, we should write $D(1^n, X_n, r)$, where r is an infinitely long random bit strong (of which D will only use a finite initial prefix). More formally, taking* Pr *to be the uniform distribution on bit-strings and over the value of X_n (or Y_n), we want*

$$|\Pr[\{r : D(1^n, X_n, r) = 1\}] - Pr[\{r : D(1^n, Y_n, r) = 1\}]| \leq \epsilon(n).$$

We similarly abuse notation elsewhere in writing Pr*.*

We often call a PPT algorithm that is supposed to distinguish between two probability ensembles a *distinguisher*.

2.2.2 Pseudorandom Functions

Definition 2.6. *A* function ensemble *is a sequence $F = \{F_n\}_{n\in\mathbb{N}}$ of random variables such that the range of F_n is the set of functions mapping n-bit strings to n-bit strings. The* uniform function ensemble*, denoted $H = \{H_n\}_{n\in\mathbb{N}}$, has H_n uniformly distributed over the set of all functions mapping n-bit strings to n-bit strings.*

We have the same notion of computational indistinguishablity for function ensembles as we had for probability ensembles, only that the distinguisher is now an oracle machine, meaning that it can query the value of the function at any point with one computation step, although it does not have the full description of the function. (See [Goldreich 2001] for a detailed description.)

We now define *pseudorandom functions* (see [Goldreich, Goldwasser, and Micali 1986]). Intuitively, this is a family of functions indexed by a seed, such that it is hard to distinguish a random member of the family from a truly randomly selected function.

Definition 2.7. *A* pseudorandom function ensemble (PRF) *is a set $\{f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ such that the following conditions hold:*

- *(easy to compute) $f_s(x)$ can be computed by a PPT algorithm that is given s and x;*
- *(pseudorandom) the function ensemble $F = \{F_n\}_{n\in\mathbb{N}}$, where F_n is uniformly distributed over the multiset $\{f_s\}_{s\in\{0,1\}^n}$, is computationally indistinguishable from H.*

We use the standard cryptographic assumption that a family of PRFs exists; this assumption is implied by the existence of one-way functions [Håstad, Impagliazzo, Levin, and Luby 1999; Goldreich, Goldwasser, and Micali 1986]. We actually require the use of a seemingly stronger notion of a PRF, which requires that an attacker getting access to polynomially many instances of a PRF (i.e., f_s for polynomially many values of s) still cannot distinguish them from polynomially many truly random functions. Nevertheless, as we show in Appendix A, it follows using a standard "hybrid" argument that any PRF satisfies also this stronger "multi-instance" security notion.

2.2.3 Public-key Encryption Schemes

We now define public-key encryption schemes. Such a scheme has two keys. The first is public and used for encrypting messages (using a randomized algorithm). The second is secret and used for decrypting. The keys are generated in such a way that the probability that a decrypted message is equal to the encrypted message is equal to 1. The key generation algorithm takes as input a "security parameter" k that is used to determine the security of the protocols (inuitively, no polynomial-time attacker should be able to "break" the security of the protocol except possibly with a probability that is a negligible function of k).

We now recall the formal definitions of public-key encryption schemes [Diffie and Hellman 1976; Rivest, Shamir, and Adleman 1978; Goldwasser and Micali 1984].

Definition 2.8. *Given a polynomial l, an l-bit* public-key encryption scheme *is a triple $\Pi = (Gen, Enc, Dec)$ of PPT algorithms where (a) Gen takes a security parameter 1^k as input and returns a (public key, private key) pair; (b) Enc takes a public key pk and a message m in a message space $\{0,1\}^{l(k)}$ as input and returns a ciphertext $Enc_{pk}(m)$; (c) Dec is a deterministic algorithm that takes a secret key sk and a ciphertext $\mathcal{C}$ as input and outputs $m' = Dec_{sk}(\mathcal{C})$, and (d)*

$$\Pr\left[\exists m \in \{0,1\}^{l(k)} \text{ such that } Dec_{sk}(Enc_{pk}(m)) \neq m\right] = 0.$$

We next define a security notion for public-key encryption. Such a security notion considers an adversary that is characterized by two PPT algorithms, A_1 and A_2. Intuitively, A_1 gets as input a public key that is part of a (public key, secret key) pair randomly generated by Gen, together with a security parameter k. A_1 then outputs two messages in $\{0,1\}^k$ (intuitively, messages it can distinguish), and some side information that it passes to A_2 (intuitively, this is information that A_2 needs, such as the messages chosen; An example of how this is used can be seen in Appendix B). A_2 gets as input the encryption of one of those messages and the side information passed on by A_1. A_2 must output which of the two messages m_0 and m_1 the encrypted message is the encryption of (where an output of $b \in \{0,1\}$ indicates that it is m_b). Since A_1 and A_2 are PPT algorithms, the output of A_2 can be viewed as a probability distribution over $\{0,1\}$. The scheme is secure if the two ensembles (i.e., the one generated by this process where the encryption of m_0 is always given to A_2, and the one where the encryption of m_1 is always given to A_2) are indistinguishable. More formally:

Definition 2.9 (Public-key security)**.** *An l-bit public-key encryption scheme $\Pi = (Gen, Enc, Dec)$ is* secure *if, for every probabilistic polynomial-time adversary $A = (A_1, A_2)$, the ensembles $\{IND_0^\Pi(A,k)\}_k$ and $\{IND_1^\Pi(A,k)\}_k$ are computationally indistinguishable, where $\{IND_b^\Pi(A,k)\}_k$ is the following PPT algorithm:*

$$\begin{aligned} IND_b^\Pi(A,k) := \quad & (pk, sk) \leftarrow Gen(1^k) \\ & (m_0, m_1, \tau) \leftarrow A_1(1^k, pk) \ (m_0, m_1 \in \{0,1\}^k) \\ & \mathcal{C} \leftarrow Enc_{pk}(m_b) \\ & o \leftarrow A_2(\mathcal{C}, \tau) \\ & \text{Output } o. \end{aligned}$$

Intuitively, the $\leftarrow$ above functions as an assignment statement, but it is not quite that, since the various algorithms are actually PPT algorithms, so their output is randomized. Formally, $IND_b^\Pi(A,k)$ is a random variable on $(\{0,1\}^*)^4$. To compute $IND_b^\Pi(A,k,r_1,r_2,r_3,r_4)$, we view r_1, r_2, r_3, and r_4 as the random bitstrings that serve as the second arguments of Gen, A_1, Enc_{pk}, and A_2, respectively. Once we add these arguments (considering, e.g., $Gen(1^k, r_1)$ and $A_1(1^k, pk, r_2)$ rather than $Gen(1^k)$ and $A_1(1^k, pk)$) these algorithms become deterministic, and $\leftarrow$ can indeed be viewed as an assignment statement.

We assume a secure public-key encryption scheme exists. We actually require a seemingly stronger notion of "multi-instance" security, where an attacker gets to see encryptions of multiple messages, each of which is encrypted using multiple keys.

Definition 2.10. *An l-bit public-key encryption scheme $\Pi = (Gen, Enc, Dec)$ is* multi-message multi-key secure *if, for all polynomials f and g, and for every probabilistic polynomial time adversary $A = (A_1, A_2)$, the ensembles $\{IND\text{-}MULT_0^{\Pi}(A, k, f, g)\}_k$ and $\{IND\text{-}MULT_1^{\Pi}(A, k, f, g)\}_k$ are computationally indistinguishable, where*

$$
\begin{array}{l}
IND\text{-}MULT_b^{\Pi}(A, k, f, g) := \\
(pk_1, sk_1) \leftarrow Gen(1^k), \ldots (pk_{g(k)}, sk_{g(k)}) \leftarrow Gen(1^k), \\
(m_0^1, \ldots, m_0^{f(k)}, m_1^1, \ldots, m_1^{f(k)}, \tau) \leftarrow A_1(1^k, pk_1, \ldots, pk_{g(k)}) \ (m_0^i, m_1^i \in \{0,1\}^k) \\
\mathcal{C} \leftarrow Enc_{pk_1}(m_b^1), \ldots, Enc_{pk_{g(k)}}(m_b^1), \ldots, Enc_{pk_1}(m_b^{f(k)}), \ldots, Enc_{pk_{g(k)}}(m_b^{f(k)}) \\
o \leftarrow A_2(\mathcal{C}, \tau) \\
Output\ o
\end{array}
$$

In this definition, there are polynomially many messages being encrypted, and each message is encrypted a polynomial number of times, using a different key each time. Other than that, the process is similar to the standard definition of security. As we show in Appendix B, any secure encryption scheme is also multi-message multi-key secure.

2.3 Commitment schemes

We now define cryptographic commitment scheme. Informally, such a scheme is a two phase two party protocol of a sender and a receiver that allows the sender to send a message to the receiver at the first phase that commits to a bit without letting the receiver get any information about that bit, and in the second phase reveals the commitment in a way that guarantees that this is the bit he committed to.

Definition 2.11. *A secure commitment scheme with perfect bindings is a pair of PPT Algorithms C and R such that:*

- *C takes as input a security parameter 1^k and a bit b and outputs (c,s), where c is a string of length k (We denote it as C_1 and call it the commitment string) and s is a string of length $k-1$ (We denote it as C_2 and call it the commitment key).*
- *R is a deterministic algorithm that gets as input two strings s and c and output $o \in \{0, 1, f\}$.*
- *The distribution $C_1(1^k, 0)$ is computationally indistinguishable from $C_1(1^k, 1)$.*
- *$R(c, s) = b$ and for all c there does not exists $s' \neq s$ such that $R(c, s') \in \{0, 1\}$.*

It is a standard assumption that such a scheme exist. This assumption is dependent on the existence of *one-way permutations* (See [Goldreich 2001] for further discussions and definitions).

3 The complexity of finding ϵ-NE in repeated games played by stateful machines

Our goal is to show that an ϵ-NE in infinitely-repeated games can be computed in polynomial time.

3.1 Preliminaries

Definition 3.1. *Let $\mathcal{G}_{a,b,c,n}$ be the set of all games with c players, at most n actions per player, integral payoffs*[5]*, maximum payoff a, and minimum payoff b.*

Note that by our assumption that the minimax payoff is 0 for all players, we can assume $a \geq 0$, $b \leq 0$, and $a - b > 0$ (otherwise $a = b = 0$, which makes the game uninteresting). We start by showing that, given a correlated strategy σ in a game G, players can get an average payoff that is arbitrarily close to their payoff in σ by playing a fixed sequence of action profiles repeatedly.

Lemma 3.2. *For all a, b, c, all polynomials q, all n, all games $G \in \mathcal{G}_{a,b,c,n}$, and all correlated strategies σ in G, if the expected payoff vector of playing σ is p then there exists a sequence sq of length $w(n)$, where $w(n) = ((a-b)q(n)+1)n^c$, such that player i's average payoff in sq is at least $p_i - 1/q(n)$.*

Proof. Given σ, we create sq the obvious way: by playing each action in proportion to the probability $\sigma(\vec{a})$. More precisely, let $r = a - b$, and define $w(n) = (rq(n)+1)n^c$, as in the statement of the lemma. We create a sequence sq by playing each action profile $\vec{a}$ $\lfloor w(n)\sigma(\vec{a})\rfloor$ times, in some fixed order. Notice that the length of this sequence is between $w(n) - n^c$ and $w(n)$. The average payoff player i gets in sq is

$$\begin{aligned} v_i' &= \frac{1}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor}\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor u_i(\vec{a}) \\ &\geq \frac{1}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor}\left(\sum_{\vec{a}\in A, u_i(\vec{a})\geq 0}(w(n)\sigma(\vec{a})-1)u_i(\vec{a}) + \sum_{\vec{a}\in A, u_i(\vec{a})<0} w(n)\sigma(\vec{a})u_i(\vec{a})\right) \\ &= \frac{w(n)\sum_{\vec{a}\in A}\sigma(\vec{a})u_i(\vec{a})}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor} - \frac{\sum_{\vec{a}\in A, u_i(\vec{a})\geq 0} u_i(\vec{a})}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor} \geq \frac{w(n)p_i}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor} - \frac{an^c}{w(n)-n^c}. \end{aligned}$$

If $p_i < 0$,

$$\begin{aligned} v_i' &\geq \frac{w(n)p_i}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor} - \frac{an^c}{w(n)-n^c} \geq \frac{w(n)p_i - an^c}{w(n)-n^c} \\ &= \frac{(rq(n)+1)n^c p_i - an^c}{(rq(n)+1)n^c - n^c} = \frac{rq(n)n^c p_i - (a-p_i)n^c}{rq(n)n^c} \geq p_i - \frac{1}{q(n)}. \end{aligned}$$

If $p_i \geq 0$,

$$\begin{aligned} v_i' &\geq \frac{w(n)p_i}{\sum_{\vec{a}\in A}\lfloor w(n)\sigma(\vec{a})\rfloor} - \frac{an^c}{w(n)-n^c} \geq p_i - \frac{an^c}{w(n)-n^c} \\ &= p_i - \frac{an^c}{(rq(n)+1)n^c - n^c} = p_i - \frac{an^c}{rq(n)n^c} \geq p_i - \frac{1}{q(n)}. \end{aligned}$$

□

Lemma 3.3. *For all a, b, c, all polynomials q and w, all $G \in \mathcal{G}_{a,b,c,n}$, and all sequences sq of length $w(n)$, if the average payoff vector of playing sq is p, then for all $\delta \leq 1/f(n)$, where $f(n) = (a-b)w(n)q(n)$, if sq is played infinitely often, player i's payoff in $G^\infty(\delta)$ is at least $p_i - 1/q(n)$.*

[5]Our result also hold for rational payoffs except then the size of the game needs to take into account the bits needed to represent the payoffs

Proof. Suppose that $sq = (a_0, \ldots, a_{w(n)-1})$, and let v_i be i's payoff from sq^∞ in $G^\infty(\delta)$. Then

$$\begin{aligned} v_i &= \delta \sum_{t=0}^{\infty} (1-\delta)^{tw(n)} \sum_{k=0}^{w(n)-1} u(a_k)(1-\delta)^k \\ &= p_i + \delta \sum_{t=0}^{\infty} (1-\delta)^{tw(n)} \sum_{k=0}^{w(n)-1} (u(a_k) - p_i)(1-\delta)^k. \end{aligned}$$

We want to bound the loss from the second part of the sum. Notice that this is a discounted sum of a sequence whose average payoff is 0. Call this sequence sq'. Observe that, because of the discounting, in the worst case, i gets all of his negative payoff in the first round of sq' and all his positive payoffs in the last round. Thus, we can bound the discounted average payoff by analyzing this case. Let the sum of i's negative payoffs in sq' be P_{neg}, which means that the sum of i's positive payoffs must be $-P_{neg}$. Let $r = a - b$, let $v_i' = \min_{\vec{a} \in A}(u_i(\vec{a}) - p_i) \geq -r$, and let $f(n) = rw(n)q(n)$, as in the statement of the lemma. So, if $\delta \leq 1/f(n)$, player $i's$ average discounted payoff in the game is at least

$$\begin{aligned} v_i &\geq p_i + \delta \sum_{t=0}^{\infty} P_{neg}(1-\delta)^{w(n)t} + (-P_{neg})(1-\delta)^{w(n)(t+1)-1} \\ &= p_i + \delta(P_{neg} + (-P_{neg})(1-\delta)^{w(n)-1}) \sum_{t=0}^{\infty} (1-\delta)^{w(n)t} \\ &= p_i + \delta(P_{neg} + (-P_{neg})(1-\delta)^{w(n)-1}) \frac{1}{1-(1-\delta)^{w(n)}} \\ &= p_i + P_{neg}\delta \frac{1-(1-\delta)^{w(n)-1}}{(1-(1-\delta)^{w(n)})} \geq p_i + \delta P_{neg} \geq p_i + \frac{P_{neg}}{f(n)} \geq p_i + \frac{v_i' w(n)}{f(n)} = p_i - 1/q(n). \end{aligned}$$

□

The next lemma shows that, for every inverse polynomial, if we "cut off" the game after some appropriately large polynomial p number of rounds (and compute the discounted utility for the finitely repeated game considering only $p(n)$ repetitions), each player's utility in the finitely repeated game is negligibly close to his utility in the infinitely repeated game—that is, the finitely repeated game is a "good" approximation of the infinitely repeated game.

Lemma 3.4. *For all a, b, c, all polynomials q, all n, all games $G \in \mathcal{G}_{a,b,c,n}$, all $0 < \delta < 1$, all strategy profiles $\vec{M}$, and all players i, i's expected utility $p_i[\vec{M}]$ in game $G^{\lceil n/\delta \rceil}(\delta)$ and $p_i[\vec{M}]$ in game $G^\infty(\delta)$ differ by at most a/e^n.*

Proof. Let $p_i^t(\vec{M})$ denote player i's expected utility if the players are playing $\vec{M}$ and the game ends at round t. Recall that $(1-\delta)^{1/\delta} \leq 1/e$.

$$\begin{aligned} & p_i^\infty(\vec{M}) - p_i^{\lceil n/\delta \rceil}(\vec{M}) \\ = \;& \delta \textstyle\sum_{t=0}^{\infty} (1-\delta)^t \sum_{h \in H^t, \vec{a} \in A} \rho_{\vec{M}}^{t+1}(h \cdot \vec{a})[u_i(\vec{a})] - \delta \sum_{t=0}^{\lceil n/\delta \rceil} (1-\delta)^t \sum_{h \in H^t, \vec{a} \in A} \rho_{\vec{M}}^{t+1}(h \cdot \vec{a})[u_i(\vec{a})] \\ = \;& \delta \textstyle\sum_{t=\lceil n/\delta \rceil + 1}^{\infty} (1-\delta)^t \sum_{h \in H^t, \vec{a} \in A} \rho_{\vec{M}}^{t+1}(h \cdot \vec{a})[u_i(\vec{a})] \\ \leq \;& \delta \textstyle\sum_{t=\lceil n/\delta \rceil}^{\infty} (1-\delta)^t a \\ = \;& \delta(1-\delta)^{\lceil n/\delta \rceil} \textstyle\sum_{t=0}^{\infty} (1-\delta)^t a = \delta(1-\delta)^{\lceil n/\delta \rceil} \frac{a}{\delta} \leq \frac{a}{e^n}. \end{aligned}$$

□

3.2 An ϵ-NE

Let $sq = (s_1, s_2 \ldots, s_m)$ be a sequence of action profiles such that the average payoff (with no discounting) of player i from playing sq repeatedly is p_i. Let $A_i^0 \subset A_i$ be a non-empty set and let $A_i^1 = A_i \setminus A_i^0$. A player can broadcast an m-bit string by using his actions for m rounds, by treating actions from A_i^0 as 0 and actions from A_i^1 as 1. Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a multi-message multi-key secure l-bit public-key encryption scheme where, if the security parameter is k, the length of an encrypted message is $z(k)$ for some polynomial z. Fix a polynomial-time pseudorandom function ensemble $\{PS_s : s \in \{0,1\}^*\}$. For a game G such that $|G| = n$, consider the following strategy σ^{NE} for player i in $G^\infty(\delta)$:

1. Play according to sq (with wraparound) as long as sq was played in the previous round.
2. After detecting a deviation by player $j \neq i$ in round t_0 (note that here we are using the assumption that players can observe the action profile played in the previous stage):
   (a) Generate a pair (pk_i, sk_i) using $\mathrm{Gen}(1^n)$. Store sk_i in memory and use the next $l(n)$ rounds to broadcast pk_i, as discussed above.
   (b) If $i = j + 1$ (with wraparound), player i does the following:
   - i records $pk_{j'}$ for all players $j' \notin \{i, j\}$ (here we are using the fact that our TMs have memory);
   - i generates a random n-bit seed $seed$;
   - for each player $j' \notin \{i, j\}$, i computes $m = Enc_{pk_{j'}}(seed)$, and uses the next $(c-2)z(n)$ rounds to communicate these strings to the players other than i and j (in some predefined order).

   (c) If $i \neq j + 1$, player i does the following:
   - i records the actions played by $j + 1$ at time slots designated for i to retrieve $Enc_{Pk_i}(seed)$;
   - i decrypts to obtain $seed$, using Dec and sk_i.
3. At a round t after the communication phase ends, the players other than j compute $PS_{seed}(t)$ and use it to determine which action profile to play according to the distribution defined by a fixed (correlated) punishment strategy against j.

Note that if the players other than j had played a punishment strategy against j, then j would get his minimax payoff of 0. What the players other than j are actually doing is playing an approximation to a punishment strategy in two senses: first they are using a psuedorandom function to generate the randomness, which means that they are not quite playing according to the actual punishment strategy. Also, j might be able to guess which pure strategy profile they are actually playing at each round, and so do better than his minimax value. As we now show, j's expected gain during the punishment phase is negligible.

Lemma 3.5. *For all a, b, c, all polynomials t and f, all n, and all games $G \in \mathcal{G}_{a,b,c,n}$, in $G^\infty(1/f(n))$, if the players other than j play σ_{-j}^{NE}, then if j deviates at round $t(n)$, j's expected payoff during the punishment phase is negligible.*

Proof. Since we want to show j's expected payoff during the punishment phase (phase (3) only) is negligible, it suffices to consider only polynomially many rounds of playing phase (3) (more precisely, at most $nf(n)$ rounds); by Lemma 3.4, any payoff beyond then is guaranteed to be negligible due to the discounting.

We construct three variants of the strategy σ_{-j}^{NE}, that vary in phases (2) and (3). We can think of these variants as interpolating between the strategy above and the use of true randomness. (These variants assume an oracle that provides appropriate information; these variants are used only to make the claims precise.)

H1 In phase (2), the punishing players send their public keys to $j+1$. For each player j' not being punished, player $j+1$ then encrypts the seed 0 using (j')'s public key, and then sends the encrypted key to j'. In phase (3), the punishing players get the output of a truly random function (from an oracle), and use it to play the true punishment strategy. (In this case, phase (2) can be eliminated.)

H2 In phase (2), the punishing players send their public keys to $j+1$. For each player j' not being punished, player $j+1$ encrypts the seed 0 using (j')'s public key, and then sends the encrypted key to j'. In phase (3), the punishing players get a joint random seed *seed* (from an oracle) and use the outputs of PS_{seed} to decide which strategy profile to play in each round. (Again, in this case, phase (2) can be eliminated.)

H3 In phase (2), the punishing players send their public keys to $j+1$. Player $j+1$ chooses a random seed *seed* and, for each player j' not being punished, $j+1$ encrypts *seed* using (j')'s public key, and then sends the encrypted key to j'. In phase (3), the punishing players use the outputs of PS_{seed} to decide which strategy profile to play in each round.

It is obvious that in $H1$, j's expected payoff is negligible. (Actually, there is a slight subtlety here. As we observed above, using linear programming, we can compute a strategy that gives the correlated minimax, which gives j an expected payoff of 0. To actually implement this correlated minimax, the players need to sample according to the minimax distribution. They cannot necessarily do this exactly (for example, 1/3 can't be computed exactly using random bits). However, given n, the distribution can be discretized to the closest rational number of the form $m/2^n$ using at most n random bits. Using such a discretized distribution, the players other than j can ensure that j gets only a negligible payoff.)

We now claim that in $H2$, j's expected payoff during the punishment phase is negligible. Assume for contradiction that a player playing $H2$ has a non-negligible payoff $\mu(n)$ for all n (i.e., there exists some polynomial $g(\cdot)$ such that $\mu(n) \geq 1/g(n)$ for infinitely many n.). Let $h(n) = n(a-b)^2(1/\mu(n))^2$. We claim that if j's expected payoff is non-negligible, then we can distinguish $h(n)$ instances of the PRF $\{PS_s : s \in \{0,1\}^n\}$ with independently generated random seeds, from $h(n)$ independent truly random functions, contradicting the multi-instance security of the PRF PS.

More precisely, we construct a distinguisher D that, given 1^n and oracle access to a set of functions $f^1, f^2, \ldots, f^{h(n)}$, proceeds as follows. It simulates H_2 (it gets the description of the machines to play as its non-uniform advice) $h(n)$ times where in iteration i', it uses the function $f^{i'}$ as the randomization source of the correlated punishment strategy. D then computes the average payoff of player j in the $h(n)$ runs, and outputs 1 if this average exceeds $\mu(n)/2$. Note that if the functions $f^1, f^2, \ldots, f^{h(n)}$ are truly independent random functions, then D perfectly simulates H_1 and thus, in each iteration i', the expected payoff of player j (during the punishment phase) is negligible. On the other hand, if the functions $f^1, f^2, \ldots, f^{h(n)}$ are $h(n)$ independent randomly chosen instances of the PRF $\{PS_s : s \in \{0,1\}^n\}$, then D perfectly simulates H_2, and thus, in each iteration i', the expected payoff of player j (during the punishment phase) is at least $\mu(n)$.

By Hoeffding's inequality [Hoeffding 1963], given m random variables $X_1, \ldots, X_m$ all of which take on values in an interval of size c', $p(|\overline{X} - E(\overline{X})| \geq r) \leq 2\exp(-\frac{2mr^2}{c'^2})$. Since, in this setting, the range of the random variables is an interval of size $a-b$, the probability that D outputs 1 when the function are truly independent is at most $2/e^n$, while the probability that D outputs 1

when the functions are independent randomly chosen instances of the PRF $\{PS_s : s \in \{0,1\}^n\}$ is at least $1 - 2/e^n$. This, in turn, means that the difference between them is not negligible, which is a contradiction. Thus, j's expected payoff in $H2$ must be negligible.

We now claim that in $H3$, player j's expected payoff during the punishment phase is also negligible. Indeed, if j can get a non-negligible payoff, then we can break the multi-message multi-key secure encryption scheme.

Again, assume for contradiction that the punished player's expected payoff in the punishment phase is a non-negligible function $\mu(n)$ for all n. We can build a distinguisher $A = (A_1, A_2)$ (which also gets the description of the machines to play as its non-uniform advice) to distinguish $\{\text{IND-MULT}_0^{\Pi}(A, n, h, c)\}_n$ and $\{\text{IND-MULT}_1^{\Pi}(A, n, h, c)\}_n$ (where we abuse notation and identify c with the constant polynomial that always returns c). Given n, A_1 randomly selects $h(n)$ messages $r_1, \ldots, r_{h(n)}$ and outputs $(0, \ldots, 0, r_1, \ldots, r_{h(n)}, (pk_1, \ldots, pk_c))$. A_2 splits its input into pieces. The first piece contains the first c encryptions in $\mathcal{C}$ (i.e., the c encryptions of the first message chosen, according to the c different encryption functions), the second the next c encryptions and so on. Notice that each piece consists of c different encryptions of the same message in both cases. It can also simulate phase (1) by following the strategy for t rounds. It then uses each piece, along with the public keys, to simulate the communication in phase (2). For piece j it uses r_j as the seed of the PRF in phase (3). It repeats this experiment for all the different pieces of the input, for a total of $h(n)$ times, and outputs 1 if the punished player's average payoff over all experiments using its strategy is more than $\mu(n)/2$.

Note that if $b = 1$, player j faces $H3$ (i.e., the distributions over runs when $b = 1$ is identical to the distribution over runs with H3, since in both cases the seed is chosen at random and the corresponding messages are selected the same way), so player j's expected payoff in the punishment phase is $\mu(n)$. Thus, by Hoeffding's inequality the probability that player j's average payoff in the punishment phase is more then $\mu(n)/2$ is $1 - 2/e^n$, so A_2 outputs 1 with that probability in the case $b = 1$. On the other hand, if $b = 0$, then this is just H_2. We know player j's expected payoff in the punishment phase in each experiment is no more than negligible in H_2, so the probability that the average payoff is more than $\mu(n)/2$ after $h(n)$ rounds, is negligible. This means that there is a non-negligible difference between the probability A outputs 1 when $b = 1$ and when $b = 0$, which contradicts the assumption that the encryption scheme is multi-message multi-key secure public key secure. Thus, the gain in $H3$ must be negligible.

$H3$ is exactly the game that the punished player faces; thus, this shows he can't hope to gain more than a negligible payoff in expectation. □

We can now state and prove our main theorem, which shows that there exists a polynomial-time algorithm for finding an ϵ-NE by showing that σ^{NE} is an ϵ-NE for all inverse polynomials ϵ, and that it can be computed in polynomial time.

Theorem 3.6. *For all a, b, c, and all polynomials q, there is a polynomial f and a polynomial-time algorithm that, for all sequences $G_1, G_2, \ldots$ of games with $G^j \in G_{a,b,c,j}$ and for all inverse polynomials $\delta \leq 1/f$, the sequence of outputs of the algorithm given the sequence $G_1, G_2, \ldots$ of inputs is a $(1/q)$-NE for $G_1^{\infty}(\delta_1), G_2^{\infty}(\delta_2), \ldots$.*

Proof. Given a game $G^n \in \mathcal{G}(a, b, c, n)$, the first step of the algorithm is to find a correlated equilibrium σ of G^n. This can be done in polynomial time using linear programming. Since the minimax value of the game is 0 for all players, all players have an expected utility of at least 0 using σ. Let $r = a - b$. By Lemma 3.2, we can construct a sequence sq of length $w(n) = (3rnq(n) + 1)n^c$ that has an average payoff for each player that is at most $1/3q(n)$ less than his payoff using σ. By Lemma 3.3, it follows that by setting the discount factor $\delta < 1/f'(n)$, where $f'(n) = 3rw(n)q(n)$, the loss due to discounting is also at most $1/3q(n)$. We can also find a punishment strategy against each player in polynomial time, using linear programming.

We can now just take σ_n^* to be the strategy σ^{NE} described earlier that uses the sequence sq, the algorithm computed, and the punishment strategies. Let $m(n)$ be the length of phase (2), including the round where the deviation occurred. (Note that $m(n)$ is a polynomial that depends only on the choice of encryption scheme—that is, it depends on l, where an l-bit public-key encryption scheme is used, and on z, where $z(k)$ is the length of encrypted messages.) Let

$$f(n) = \max(3q(n)(m(n)a + 1), f'(n)).$$

Note that f is independent of the actual game, as required.

We now show that $\sigma_1^*, \ldots$ as defined above is a $(1/q)$-NE. If in game G^n a player follows σ_n^*, he gets at least $-2/3q(n)$. Suppose that player j defects at round t; that is, that he plays according to σ_n^* until round t, and then defects. By Lemma 3.4 if $t > \frac{n}{\delta(n)}$, then any gain from defection is negligible, so there exists some n_1 such that, for all $n > n_1$, a defection in round t cannot result in the player gaining more than $\frac{1}{q(n)}$. If player j defects at round $t \leq \frac{n}{\delta(n)}$, he gets at most a for the duration of phase (2), which is at most $m(n)$ rounds, and then, by Lemma 3.5, gains only a negligible amount, say $\epsilon_{neg}(n)$ (which may depend on the sequence of deviations), in phase (3). Let u_i^n be the payoff of player i in game G^n of the sequence. It suffices to show that

$$\delta(n)(\sum_{k=0}^{t} u_i^n(a_k)(1-\delta(n))^k + \sum_{k=0}^{m(n)} a(1-\delta(n))^{k+t} + (1-\delta(n))^{t+m(n)}\epsilon_{neg}(n)) - 1/q(n) \leq$$
$$\delta(n)(\sum_{k=0}^{t} u_i^n(a_k)(1-\delta(n))^k + \sum_{k=t}^{\infty} u_i^n(a_k)(1-\delta(n))^k).$$

By deleting the common terms from both side, rearranging, and noticing that $(1-\delta(n))^{m(n)}\epsilon_{neg(n)} \leq \epsilon_{neg(n)}$, it follows that it suffices to show

$$\delta(n)(1-\delta(n))^t(\sum_{k=0}^{m(n)} a(1-\delta(n))^k + \epsilon_{neg}(n)) - \frac{1}{q(n)} \leq \delta(n)(1-\delta(n))^t(\sum_{k=0}^{\infty} u_i^n(a_{k+t})(1-\delta(n))^k).$$

We divide both sides of the equation by $(1-\delta(n))^t$. No matter at what step of the sequence the defection happens, the future expected discounted payoff from that point on is still at least $-2/3q(n)$, as our bound applies for the worst sequence for a player, and we assumed that in equilibrium all players get at least 0. Also notice that $(1-\delta(n))^t < 1$. It follows that we need to show that

$$\delta(n)(\sum_{k=0}^{m(n)} a(1-\delta(n))^k + \epsilon_{neg}(n)) \leq \frac{1}{3q(n)}.$$

This means that a player wants to defect at some round t only if he already wants to defect at the first round, as this is exactly the equation that we get for a defection at the first round. Since ϵ_{neg} is negligible for all deviations, it follows that, for all sequences of deviations, there exists n_0 such

that $\epsilon_{neg}(n) < 1$ for all $n \geq n_0$. For $n \geq n_0$,

$$\begin{aligned}\delta(n)(\sum_{k=0}^{m(n)} a(1-\delta(n))^k + \epsilon_{neg}(n)) &\leq \delta(n)(m(n)a + \epsilon_{neg}(n)) \\ &\leq \frac{m(n)a + \epsilon_{neg}(n)}{f(n)} \\ &\leq \frac{m(n)a + \epsilon_{neg}(n)}{3q(n)(m(n)a+1)} \\ &\leq \frac{1}{3q(n)}.\end{aligned}$$

This shows that there is no deviating strategy that can result in the player gaining more than $\frac{1}{q(n)}$ in G^n for $n > \max\{n_0, n_1\}$. □

4 Subgame-Perfect Equilibrium

In this section, we show that a similar approach allows us to compute a subgame-perfect ϵ-equilibrium (for the original definition of subgame-perfect equilibrium see [Selten 1965; Selten 1975]). This equilibrium concept requires that the strategies form an ϵ-NE after every possible deviation of players. The intuition is that if this is not true, our punishment strategies will not be credible threats, since a player will not want to punish when his strategy says that he should. In the standard, non-computational, setting, this requires that the strategies are an ϵ-NE at any history of the game, even histories that are not on any equilibrium path. However, since we consider stateful TMs, there is more to a description of a situation that just the history; we need to know the memory state of the TM. A deviating strategy TM can change its memory state in arbitrary ways, so when we argue that switching to the equilibrium strategies is an ϵ-NE in a history, we must also consider all possible states that the TM might start with at that history. Since there exists a deviation that just rewrites the memory in the step just before the history we are considering, any memory state (of polynomial length) is possible. For these reasons, in the computational setting we require that the TMs strategies are an ϵ-NE at every history, no matter what the states of the TMs are at that history.

One consequence of this requirement is that even when a TM is expected to play honestly it sometimes can't, since its memory state lacks the information it needs to do so. For example, it may not be able to decrypt a message if its memory does not contain the key required. This means that a strategy must describe what to do even in situations where the current memory state does not match what it "should" be, given the history.

As a player's TM can not observe the memory state of the other players' TMs, and we do not assumes the players know what exact defection strategy was played, the game is in some sense an imperfect information game; the same history can result in very different memory states and thus different states of the game are in the same information set of a player. In such games, it is common to consider *sequential equilibrium* [Kreps and Wilson 1982] as the desired solution concept. This solution concept specifies in addition to a strategy at every history also a belief at every history (with some consistency requirements) and the strategy needs to be a best response given this belief. Our notion of sub-game perfection can be viewed as a very strong version of a sequential equilibrium. We require the strategy to be a best response (up to ϵ) independent of the belief of the player about the other players memory state.

Let H_G be the set of all histories in a game G. For $h \in H_G$, let G^h be the subgame of G starting from history h. For a memory state m and a TM M let $M(m)$, stand for running M with initial

memory state m. We use $\vec{M}(\vec{m})$ to denote $(M_1(m_1), \ldots, M_c(m_c))$. We now define subgame-perfect equilibrium in our computational setting. As before, this requires considering a sequence of games rather than a single game.

Definition 4.1. *An infinite sequence of strategy profiles $\vec{M}^1, \vec{M}^2, \ldots$, where $\vec{M}^k = (M_1^k, ..., M_c^k)$, is a subgame-perfect ϵ-equilibrium of an infinite sequence of repeated games $G_1^\infty, G_2^\infty, \ldots$ where the size of G_k is k, if, for all players $i \in [c]$, all possible sequences of histories $h_1 \in H_{G_1^\infty}, h_2 \in H_{G_2^\infty}, \ldots$, all possible sequences of polynomial length memory state profiles $\vec{m}^1, \vec{m}^2, \ldots$, where $\vec{m}^k = (m_1^k, \ldots, m_c^k)$, and all infinite sequences of polynomial-time TMs $M^1, M^2, \ldots$ (polynomial in n and t, as discussed above), there exists k_0 such that, for all $k \geq k_0$,*

$$p_i^k(M^k(m_i^k), \vec{M}_{-i}^k(\vec{m}_{-i}^k)) \leq p_i^k(\vec{M}^k(\vec{m}^k)) + \epsilon(k),$$

where p_i^k is the payoff of player i in game $(G_k^{h_k})^\infty$.

The strategy σ^{NE} defined above is not a subgame-perfect equilibrium. Once one of the players deviates and must be punished, the punishing players might get much less than they would if they did not carry out the punishment. Thus, in a history where they are supposed to punish other players according to their strategy (which must be a history off the equilibrium path), they might want to deviate. For a game G such that $|G| = n$, consider instead the strategy $\sigma^{NE^*,q}$ for player i in $G^\infty(\delta)$, in which phases (1) and (2) are the same, but phase (3) is played for only $q(n)$ rounds, after which the players go back to playing according to (1) starting from the same place in the sequence where they left off when the deviation occurred.

As the TM's action in phase (1) does not depend on its memory state, we can say that $\sigma^{NE^*,q}$ plays phase (1) no matter what its saved in its memory. In phases (2) and (3), if it already sent the public key but does not remember the public key, it plays a fixed action until the end of phase (3), at which point it continues from phase (1) again. Notice, that it easy to compute from a history what phase is being played and, since the lengths are fixed, this strategy is easy to implement.

We now state and prove our theorem, which shows that there exists a polynomial-time algorithm for computing a subgame-perfect ϵ-equilibrium by showing that, for all inverse polynomials ϵ, $\sigma^{NE^*,\ell}$, for some function ℓ of ϵ, is a subgame-perfect ϵ-equilibrium of the game .

Theorem 4.2. *For all a, b, c, and all polynomials q, there is a polynomial f and a polynomial-time algorithm F such that, for all sequences $G_1, G_2, \ldots$ of games with $G^j \in G_{a,b,c,j}$ and for all inverse polynomials $\delta \leq 1/f$, the sequence of outputs of F given the sequence $G_1, G_2, \ldots$ of inputs is a subgame-perfect $\frac{1}{q}$-equilibrium for $G_1^\infty(\delta_1), G_2^\infty(\delta_2), \ldots$.*

Proof. The algorithm is similar to the one described in Theorem 3.6. It computes the sequence to be played and the punishment strategy in the same way. Let m and f' be the same as in the proof of Theorem 3.6.

Let $\ell(n) = nq(n)(m(n)a + 1)$, let σ_n^* to be the strategy $\sigma^{NE^*,\ell}$ described above, let $r = a - b$, and let $f(n) = \max(3rq(n)(\ell(n) + m(n)), f'(n))$. We now show that $\sigma_1^*, \sigma_2^*, \ldots$ is a subgame-perfect $(1/q)$-equilibrium for every inverse polynomial discount factor $\delta \leq 1/f$.

As before, we can focus only on rounds $t < \frac{n}{\delta(n)}$, since, by Lemma 3.4, the sum of payoffs received after that is negligible. Thus, there exists some n_0 such that, for all $n > n_0$, the payoff achieved after that round is less than $1/q(n)$. Such a payoff cannot justify a defection.

We first show that no player has an incentive to deviate in subgames starting from histories where phase (1) is being played. As in the proof of Theorem 3.6, since sq is played both at round t and at round 1, it is easy to see that a defection strategy that increases player i's utility at round t can be modified to a defection strategy that increases player i's utility at round 1, so we consider only defections in the first round. Also notice that if it is profitable for a player to deviate, it will

also be profitable to deviate after the punishment phase when the players get back to playing phase (1). The payoff during one cycle of a defection and punishment can be at most a while in phase (2) and then negligible throughout phase (3). This means that it suffices to prove that

$$\delta(n)(m(n)a + \epsilon_{neg}) \sum_{t=0}^{\infty} (1 - \delta(n))^{(m(n)+\ell(n))t} \leq \frac{1}{3q(n)}.$$

The term on the left side is bounded by $O\big(\frac{m(n)a+\epsilon_{neg}}{nq(n)(m(n)a+1)}\big)$, and thus there exists n_1 such that, for all $n > n_1$, the term on the left side is smaller than $\frac{1}{3q(n)}$.

We next show that no player wants to deviate at another history. First consider the punishing player. By not following the strategy, he can gain at most r for at most $\ell(n) + m(n)$ rounds over the payoff he gets with the original strategy (this is true even if his memory state is such that he just plays a fixed action). Once the players start playing phase (1) again, our previous claim shows the player does not have an incentive to deviate. It is easy to verify that, given the discount factor, a deviation can increase his discounted payoff by at most $\frac{1}{q(n)}$ in this case.

The punished player can defect to a Turing machine that correctly guesses the keys chosen (or the current TM's memory state contains the actual keys and he defects to a TM that uses these keys) and thus knows exactly what the players are going to do while they are punishing him. Such a deviation exists once the keys have been played and are part of the history. Another deviation might be a result of the other TMs being in an inconsistent memory state, so that they play a fixed action, one which the deviating player might be able to take advantage of. However, these deviations work only for the current punishment phase. Once the players go back to playing phase (1), this player does not want to deviate again. For if he deviates again, the other players will choose new random keys and a new random seed (and will have a consistent memory state), so his expected value from future deviations is the same as his expected value if he does not deviate. This means he can also gain at most r for at most $\ell(n) + m(n)$ rounds which, as claimed before, means that his discounted payoff difference is less than $\frac{1}{q(n)}$ in this case.

This shows that, for n sufficiently large, no player can gain more than $1/q(n)$ from defecting at any history. Thus, this strategy is indeed a subgame-perfect $1/q$-equilibrium. □

5 Variable-player games

In this section we show that our techniques can also be used to find a subgame-perfect ϵ-NE in repeated games with a variable number of players (i.e., games where the number of players is part of the input and not fixed). In general, just describing players' utilities in such a game takes space exponential in the number of players (since there are exponentially many strategy profiles). Thus, to get interesting computational results, we consider games that can be represented succinctly. Specifically, we focus on a family of multi-player games called *graphical games of degree at most d*, for some d. These are game that can be represented by a graph in which each player is a node in the graph, and the utility of a player is a function of only his action and the actions of the players to which he is connected by an edge. The maximum degree of a node is assumed to be at most d. This means a player's punishment strategy depends only on the actions of at most d players.

Definition 5.1. *Let $\mathcal{G}'_{a,b,d,n,m}$ be the set of all graphical games with degree at most d, at most m players and at most n actions per player, integral payoffs,*[6] *maximum payoff a, and minimum payoff b.*

[6]Our result also hold for rational payoffs, except then the size of the game needs to take into account the bits needed to represent the payoffs.

Papadimitriou and Roughgarden [2008] show that for many families of succinctly represented variable-player games, which includes graphical games, there is a polynomial-time algorithm for finding a correlated equilibrium of the game. The structure of this correlated equilibrium is a convex combination of product distributions over players' actions of size polynomial in the number of players and actions. It is not clear how to use our previous technique of converting the correlated equilibrium to a deterministic polynomial-length sequence, as the support of this equilibrium might be of exponential size.

For a game $G \in \mathcal{G}'_{a,b,d,n,n}$, consider the strategy $\sigma^{NE^*,\ell_1,\ell_2}$, which is again a three phase strategy. Phase (1) is split into two parts. In phase (1a), two predefined players (which we call the committing player and the receiving player) play the following protocol $n(n+1)$ times: The committing player chooses uniformly at random a bit and uses a commitment scheme to compute a commitment string for that bit, and then broadcast it using its actions. The receiving player then chooses uniformly at random a bit and broadcast it using its actions. After that, the committing player broadcast the commitment key. The players then xor the two bits. After the $n(n+1)$ iteration the players have a string of length $n(n+1)$ which they can use as a random seed. Since all communication is public, at the end of this phase, all the players know the joint secret seed. In phase (1b), this seed is used as the seed for a pseudo-random function whose outputs are used as the random bits to sample from the correlated equilibrium for $\ell_2(n)$ rounds, where ℓ_2 is a polynomial. (Note that it is possible to sample a strategy of this equilibrium using this many bit of randomness.) The players repeatedly iterate phases (1a) and (1b). Every time the committing player sends an invalid commitment key during any iteration of phase (1a), the players play a fixed action through phase (1b). When the phase is over, the players go back to playing phase (1a), but switch the roles of the two predefined players from that point on (at least until the new committing player sends an invalid key in which case they switch back). If a player deviates in phase (1b) (notice that this is observable by the players since once the seed is chosen the PRF outputs describe a deterministic sequence), the players punish him by playing phases (2) and (3). When the punishment phase is over the player go back to play (1b) from the point of the deviation. Phase (2) and (3) are the same as in σ^{NE^*,ℓ_1} (although the punishment is only carried out by the players that influence the punished player utility). If at any point the memory state is not consistent with the history played, the TM plays some fixed action, until the relevant phase is over and it can play according to the strategy again.

Lemma 5.2. *For all $G \in \mathcal{G}'_{a,b,d,n,n}$, and for all polynomials q, if η is a correlated equilibrium of G of the form described above, then there exist polynomials ℓ_2 and f such that for all players i, at every round of the game its expected payoff from playing $\sigma^{NE^*,\ell_1,\ell_2}$ based on η in $G^\infty(\frac{1}{f(n)})$, given the rest of the players are also playing $\sigma^{NE^*,\ell_1,\ell_2}$, is at least $-\frac{1}{q(n)}$.*

Proof. Let $v(n)$ be the length of phase (1a). We first show that the expected value from one iteration of phase (1a) and (1b) is at least $bv(n) - \epsilon_{neg}$ for some negligible function ϵ_{neg}. It is obvious that from phase (1a) they get at least $bv(n)$, so we need to show that at the start of phase (1) their expected value from phase (1b) is at least $-\epsilon_{neg}$. We just sketch the proof, since it is similar to that of Lemma 3.5.

First notice that if all the players follow the strategy as in the Lemma statement, the seed generated at phase (1a) is uniformly distributed over $\{0,1\}^{n(n+1)}$. We construct two variants of the game. The first variant is such that the players use the output of a truly random function (which they get from an oracle) to play the correlated equilibrium. In the second, they use a truly random seed (from phase (1a)) as the seed of a PRF and use its outputs to play the correlated equilibrium.

Just as in the proof of Lemma 3.5, it is easy to verify that in the first variant, their payoff decreases by a negligible function (due to the use of only polynomially many random bits, just as in the argument about H1 in the proof of Lemma 3.5). If the second variant does not differ only

negligibly from the first, then we can use it to break the PRF. We leave the rest of the details to the reader.

It is also easy to verify that the players get at least $bm(n)-\epsilon_{neg}$ in each of the first polynomially many repetition of phase (1) (for reasons similar to the one used to prove Lemma A.1). Thus, we know that the expected payoff of the player in round 0 is

$$\frac{1}{f(n)}(v(n)b-\epsilon_{neg})\sum_{t=0}^{nf(n)}(1-\frac{1}{f(n)})^{(v(n)+\ell_2(n))t}-\epsilon'_{neg},$$

where ϵ'_{neg} represents the player's payoff after $nf(n)(v(n)+\ell_2(n))$ rounds, and must be negligible by Lemma 3.4.

It is easy to verify that at any round t a player's payoff is then at least

$$\frac{1}{f(n)}\big(b(v(n)+\ell_2(n))+(v(n)b-\epsilon_{neg})\sum_{t=0}^{nf(n)}(1-\frac{1}{f(n)})^{(v(n)+\ell_2(n))t}\big)-\epsilon'_{neg},$$

since in one iteration of phase (1), the least the player can gain is $b(v(n)+\ell_2(n))$ and his expected value for future iterations is the same as in round 0.

Let $\ell_2 = nq(n)(-v(n)b+1)$ and let $f(n) = nq(n)(v(n)+\ell_2(n))$. Similarly to the proof of Theorem 4.2, this choice of ℓ_2 and f guarantees that for a large enough n we get our desired result. □

We now state and prove our theorem, which shows that there exists a polynomial-time algorithm for computing a subgame-perfect ϵ-equilibrium by showing that, for all inverse polynomials ϵ, $\sigma^{NE^*,\ell_1,\ell_2}$, for some functions ℓ_1,ℓ_2 of ϵ, is a subgame-perfect ϵ-equilibrium of the game .

Theorem 5.3. *For all a, b, d, and all polynomials q, there is a polynomial f and a polynomial-time algorithm such that, for all sequences $G_1, G_2, \ldots$ of games with $G^j \in G_{a,b,d,j,j}$ and for all inverse polynomials $\delta \le 1/f$, the sequence of outputs of the algorithm given the sequence $G_1, G_2, \ldots$ of inputs is a subgame-perfect $\frac{1}{q}$-equilibrium for $G_1^\infty(\delta_1), G_2^\infty(\delta_2), \ldots$.*

Proof. Again, we just provide a sketch. Let ℓ_1 be as in the proof of Theorem 4.2 and let ℓ_2 be as in the proof of Lemma 5.2. Let f be the max of f from Theorem 4.2 and f from Lemma 5.2. By Lemma 5.2 the expected payoff in any round of the game is at least $-\frac{1}{2q(n)}$. Any deviation at a history where phase (1b), (2) or (3) (even such where the memory state is inconsistent) is played is similar to the deviation considered in the proof of Theorem 4.2. This means that similarly to that proof, the expected discounted payoff from any such deviation is less than $\frac{1}{2q(n)}$, which means that by defecting at these histories the player can't gain more than $\frac{1}{q(n)}$.

We now focus on deviations in histories where phase (1a) is played. We first notice that the receiving player can't make phase (1a) fail. Any action he plays during that phase is a legal action. It is also easy to verify that he can only bias the random string with negligible probability (it is obvious that at any iteration of a bit commitment he can bias the resulting bit with negligible probability and similar ideas to the ones in the proof of Lemma A.1 can be used to show that the same hold for polynomially many such iterations). Using an hybrid argument similar to the one used in Lemma 5.2 (with an extra phase where the PRF uses a real random string), this can be shown to mean that this player can't get more than negligible over his expected payoff and thus such a deviation is not profitable.

Next we notice that the committing player can only cause the protocol to fail by sending a invalid commitment key, but he can't bias a legal run of the protocol. If the players does send an illegal key, he might gain r for one run of phase (1), but after that he will be playing the receiving

player role and by our previous argument could not gain more than negligible from that point on. Thus, as in Lemma 5.2 the chosen discount factor guarantees that this can't gain him enough to make such deviation justified. Notice that the same reasoning shows that if its memory state is not consistent with the commitment and he can't send the right key, no defection is profitable enough to make him not play according to the specified action.

This shows that there is no defection that gains more than $frac1q(n)$ and thus we get the desired result. □

6 Conclusions

We have shown that computing an ϵ-NE for an infinitely-repeated game is indeed easier than computing one in a normal-form game. Our techniques use threats and punishment much in the same way that they are used to prove the Folk Theorem. However, there is a new twist to our results. We must assume that players are computationally bounded, and we use some cryptographic techniques that rely on standard cryptographic assumptions.

7 Acknowledgments

Joseph Halpern and Lior Seeman are supported in part by NSF grants IIS-0812045, IIS-0911036, and CCF-1214844, by AFOSR grant FA9550-08-1-0266, by ARO grant W911NF-09-1-0281, and by the Multidisciplinary University Research Initiative (MURI) program administered by the AFOSR under grant FA9550-12-1-0040. Rafael Pass is supported in part by an Alfred P. Sloan Fellowship, a Microsoft Research Faculty Fellowship, NSF Awards CNS-1217821 and CCF-1214844, NSF CAREER Award CCF-0746990, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

APPENDIX

A Multi-Instance PRFs

In this section, we show that for any family of PRF, even polynomially many random members of it are indistinguishable from polynomially many truly random functions.

Lemma A.1. *For all polynomials q, if $\{f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ is a pseudorandom function ensemble, then the ensemble $F^q = \{F_n^1, \ldots, F_n^{q(n)}\}_{n\in\mathbb{N}}$ where, for all i, F_n^i is uniformly distributed over the multiset $\{f_s\}_{s\in\{0,1\}^n}$, is computationally indistinguishable from $H^q = \{H_n^1, \ldots, H_n^{q(n)}\}_{n\in\mathbb{N}}$.*

Proof. Assume for contradiction that the ensembles are distinguishable. This means there exist a polynomial q, a PPT D, and a polynomial p such that for infinitely many n's

$$|Pr[D(1^n, (H_n^1, \ldots, H_n^{q(n)})) = 1] - Pr[D(1^n, (F_n^1, \ldots, F_n^{q(n)})) = 1]| > \frac{1}{p(n)}.$$

For each n, let $T_n^i = (1^n, (H_n^1, \ldots, H_n^{i-1}, F_n^i, \ldots, F_n^{q(n)}))$. We can now describe a PPT D' that distinguishes $\{F_n\}_{n\in\mathbb{N}}$ and $\{H_n\}_{n\in\mathbb{N}}$ for infinitely many n's. First notice that a PPT can easily simulate polynomially many oracle queries to both a truly random function and to a member of F_n. So D' on input $(1^n, X)$ randomly chooses $j \in \{1, \ldots, q(n)\}$ and calls D with input

$(1^n, (I^1, \ldots, I^{j-1}, X, J^{j+1}, \ldots, J^{q(n)}))$, where it simulates a query to I_k as a query to a random member of H_n, and a query to J_k as a query to a random member of F_n. (Notice that since D is a PPT, it can make only polynomially many oracle queries to any of the functions, which can be easily simulated). Whenever D makes an oracle query to X, D' makes an oracle query to X, and uses its answer as the answer to D. When D terminates, D' outputs the same value as D.

Now notice that if X is H_n, then the input to D is T_n^j, while if X is F_n, then the input to D is T_n^{j+1}. Thus, $Pr[D'(1^n, H_n) = 1] = \frac{1}{q(n)} \sum_{i=1}^{q(n)} Pr[D(T_n^{i+1}) = 1]$, and
$Pr[D'(1^n, F_n) = 1] = \frac{1}{q(n)} \sum_{i=1}^{q(n)} Pr[D(T_n^i) = 1]$. It follows that

$$\begin{aligned}
|Pr[D'(1^n, H_n) = 1] - Pr[D'(1^n, F_n) = 1]| &= \frac{1}{q(n)} |\sum_{i=1}^{q(n)} Pr[D(T_n^{i+1}) = 1] - Pr[D(T_n^i) = 1]| \\
&= \frac{1}{q(n)} |Pr[D(T_n^{q(n)+1}) = 1] - Pr[D(T_n^1) = 1]| \\
&> \frac{1}{q(n)p(n)},
\end{aligned}$$

where the last inequality is due to the fact that $T_n^{q(n)+1} = (1^n, (H_n^1, \ldots, H_n^{q(n)}))$ and $T_n^1 = (1^n, (F_n^1, \ldots, F_n^{q(n)}))$. But this means that for any such n, D' can distinguish $F = \{F_n\}_{n\in\mathbb{N}}$ and $H = \{H_n\}_{n\in\mathbb{N}}$ with non-negligible probability, and thus can do that for infinitely many n's. This is a contradiction to the assumption that $\{f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ is a pseudorandom function ensemble. □

B Multi-key Multi-Message Security

In this section, we show that any secure public-key encryption scheme is also multi-key multi-message secure.

Lemma B.1. *If (Gen, Enc, Dec) is a secure public key encryption scheme, then it is also multi-message multi-key secure.*

Proof. Assume for contradiction that $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is a secure public key encryption scheme that is not multi-message multi-key secure. Then there exist polynomials f and g and an adversary $A = (A_1, A_2)$ such that $\{\text{IND-MULT}_0^{\Pi}(A, k, f, g)\}_k$ and $\{\text{IND-MULT}_1^{\Pi}(A, k, f, g)\}_k$ are distinguishable. That means there exist a PPT D and a polynomial p such that

$$|Pr[D(1^k, \{\text{IND-MULT}_0^{\Pi}(A, k, f, g)\}) = 1] - Pr[D(1^k, \{\text{IND-MULT}_1^{\Pi}(A, k, f, g)\}) = 1]| > \frac{1}{p(n)}.$$

Let $T_{i,j}^{\pi}(A, k, f, g)$ be the following PPT algorithm:

$$\begin{aligned}
T_{i,j}^{\pi}(A, k, f, g) := \quad & (pk_1, sk_1) \leftarrow \mathrm{Gen}(1^k), \ldots (pk_{g(k)}, sk_{g(k)}) \leftarrow \mathrm{Gen}(1^k), \\
& (m_0^1, \ldots, m_0^{f(k)}, m_1^1, \ldots, m_1^{f(k)}, \tau) \leftarrow A_1(1^k, pk_1, \ldots, pk_{g(k)}) \\
& \mathcal{C} \leftarrow \mathrm{Enc}_{pk_1}(m_0^1), \ldots, \mathrm{Enc}_{pk_{g(k)}}(m_0^1), \\
& \ldots, \mathrm{Enc}_{pk_1}(m_0^j), \ldots, \mathrm{Enc}_{pk_{i-1}}(m_0^j), \mathrm{Enc}_{pk_i}(m_1^j), \ldots \mathrm{Enc}_{pk_{g(k)}}(m_1^j), \\
& \ldots \mathrm{Enc}_{pk_1}(m_1^{f(k)}), \ldots, \mathrm{Enc}_{pk_{g(k)}}(m_1^{f(k)}) \\
& o \leftarrow A_2(\mathcal{C}, \tau) \\
& \text{Output } o.
\end{aligned}$$

We now define an adversary $A' = (A'_1, A'_2)$, and show that $\{\text{IND}_0^\Pi(A', k, f, g)\}_k$ and $\{\text{IND}_1^\Pi(A', k, f, g)\}_k$ are not computationally indistinguishable. A'_1 on input $(1^k, pk)$ first chooses $i \in \{1, \ldots, g(k)\}$ uniformly at random. It then generates $g(k) - 1$ random key pairs $(pk_1, sk_1), \ldots, (pk_{i-1}, sk_{i-1}), (pk_{i+1}, sk_{i+1}), \ldots, (pk_{g(k)}, sk_{g(k)})$. It then calls A_1 with input $(1^k, pk_1, \ldots, pk_{i-1}, pk, pk_{i+1}, \ldots, pk_{g(k)})$. After getting A_1's output $M = (m_0^1, \ldots, m_0^{f(k)}, m_1^1, \ldots, m_1^{f(k)}, \tau)$, A'_1 chooses $j \in \{1, \ldots, f(n)\}$ uniformly at random, and returns as its output $(m_0^j, m_1^j, (i, j, pk, pk_1, sk_1, \ldots, pk_{g(k)}, sk_{g(k)}, M))$.

A'_2 on input $(\mathcal{C}, (i, j, pk, pk_1, sk_1, \ldots, pk_{g(k)}, sk_{g(k)}, M))$ constructs input $\mathcal{C}'$ for A_2 by first appending the encryptions of messages $m_0^1 \ldots, m_0^{j-1}$ with all the keys, then appending the encryption of m_0^j with keys $pk_1, \ldots, pk_i$ and then appends $\mathcal{C}$. It then appends the encryption of m_1^j with keys $pk_{i+2}, \ldots, pk_{g(k)}$ and also the encryption of the messages $m_1^{j+1}, \ldots, m_1^{f(k)}$ with each of the keys. It then outputs $A_2(\mathcal{C}', \tau)$. If $\mathcal{C}$ is the encryption of m_j^0 with key pk, then this algorithm is identical to $T^\pi_{i+1,j}(A, k, f, g)$ (if $i = g(k)$ then by $T^\pi_{i+1,j}$ we mean $T^\pi_{1,j+1}$; we use similar conventions elsewhere), while if it is the encryption of m_j^1 with key pk, then the algorithm is identical to $T^\pi_{i,j}(A, k, f, g)$.

We claim that D can distinguish $\{\text{IND}_0^\Pi(A', k, f, g)\}_k$ and $\{\text{IND}_1^\Pi(A', k, f, g)\}_k$. Note that

$$Pr[D(1^k, \{\text{IND}_0^\Pi(A', k, f, g)\}) = 1] = \frac{1}{g(k)f(k)} \sum_{j=1}^{f(k)} \sum_{i=1}^{g(k)} Pr[D(1^k, T^\pi_{i+1,j}(A, k, f, g)) = 1]$$

and

$$Pr[D(1^k, \{\text{IND}_1^\Pi(A', k, f, g)\}) = 1] = \frac{1}{g(k)f(k)} \sum_{j=1}^{f(k)} \sum_{i=1}^{g(k)} Pr[D(1^k, T^\pi_{i,j}(A, k, f, g)) = 1].$$

Thus,

$$\begin{aligned}
& |Pr[D(1^k, \{\text{IND}_0^\Pi(A', k, f, g)\}) = 1] - Pr[D(1^k, \{\text{IND}_1^\Pi(A', k, f, g)\}) = 1]| \\
= \; & \tfrac{1}{g(k)f(k)} |\textstyle\sum_{j=1}^{f(k)} \sum_{i=1}^{g(k)} (Pr[D(1^k, T^\pi_{i+1,j}(A, k, f, g)) = 1] - Pr[D(1^k, T^\pi_{i,j}(A, k, f, g)) = 1]) \\
= \; & \tfrac{1}{g(k)f(k)} |Pr[D(1^k, \{\text{IND-MULT}_0^\Pi(A, k, f, g)\}) = 1] - Pr[D(1^k, \{\text{IND-MULT}_1^\Pi(A, k, f, g)\}) = 1]| \\
> \; & \tfrac{1}{g(k)f(k)p(k)},
\end{aligned}$$

where the next-to-last line follows because $T^\pi_{1,1}(A, k, f, g) = \text{IND-MULT}_1^\Pi(A, k, f, g)$ and $T^\pi_{g(k)+1,f(k)}(A, k, f, g) = \text{IND-MULT}_0^\Pi(A, k, f, g)$. Thus, we have a contradiction to the fact that the encryption scheme is secure. □

References

Andersen, G. and V. Conitzer (2013). Fast equilibrium computation for infinitely repeated games. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Borgs, C., J. Chayes, N. Immorlica, A. Kalai, V. Mirrokni, and C. Papadimitriou (2010). The myth of the folk theorem. *Games and Economic Behavior 70*(1), 34–43.

Chen, X. and X. Deng (2006). Settling the complexity of two-player nash equilibrium. In *Proc. 47th IEEE Symposium on Foundations of Computer Science*, pp. 261–272.

Chen, X., X. Deng, and S. Teng (2006). Computing Nash equilibria: Approximation and smoothed complexity. In *Proc. 47th IEEE Symposium on Foundations of Computer Science*, pp. 603–612.

Daskalakis, C., P. Goldberg, and C. Papadimitriou (2006). The complexity of computing a nash equilibrium. In *Proc. 38th ACM Symposium on Theory of Computing*, pp. 71–78.

Diffie, W. and M. Hellman (1976). New directions in cryptography. *IEEE Transactions on Information Theory 22*(6), 644–654.

Dodis, Y., S. Halevi, and T. Rabin (2000). A cryptographic solution to a game theoretic problem. In *CRYPTO 2000: 20th International Cryptology Conference*, pp. 112–130.

Goldreich, O. (2001). *Foundation of Cryptography, Volume I Basic Tools*.

Goldreich, O., S. Goldwasser, and S. Micali (1986). How to construct random functions. *Journal of the ACM 33*(4), 792–807.

Goldwasser, S. and S. Micali (1984). Probabilistic encryption. *Journal of Computer and System Sciences 28*(2), 270–299.

Gossner, O. (1998). *Repeated games played by cryptographically sophisticated players*. Center for Operations Research & Econometrics. Université catholique de Louvain.

Gossner, O. (2000). Sharing a long secret in a few public words. Technical report, THEMA (THéorie Economique, Modélisation et Applications), Université de Cergy-Pontoise.

Håstad, J., R. Impagliazzo, L. A. Levin, and M. Luby (1999). A pseudorandom generator from any one-way function. *SIAM Journal on Computing 28*(4), 1364–1396.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*(301), 13–30.

Kreps, D. M. and R. Wilson (1982). Sequential equilibria. *Econometrica: Journal of the Econometric Society*, 863–894.

Lehrer, E. (1991). Internal correlation in repeated games. *International Journal of Game Theory 19*(4), 431–456.

Littman, M. L. and P. Stone (2005). A polynomial-time nash equilibrium algorithm for repeated games. *Decision Support Systems 39*(1), 55–66.

Neyman, A. (1985). Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters 19*(3), 227–229.

Osborne, M. J. and A. Rubinstein (1994). *A Course in Game Theory*. Cambridge, Mass.: MIT Press.

Papadimitriou, C. H. and T. Roughgarden (2008). Computing correlated equilibria in multiplayer games. *Journal of the ACM (JACM) 55*(3), 14.

Papadimitriou, C. H. and M. Yannakakis (1994). On complexity as bounded rationality. In *Proc. 26th ACM Symposium on Theory of Computing*, pp. 726–733.

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

Rubinstein, A. (1986). Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory 39*(1), 83–96.

Selten, R. (1965). Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit: Teil i: Bestimmung des dynamischen preisgleichgewichts. *Zeitschrift für die gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics 121*(2), 301–324.

Selten, R. (1975). Reexamination of the perfectness concept for equilibrium points in extensive games. *International journal of game theory 4*(1), 25–55.

Urbano, A. and J. Vila (2004). Unmediated communication in repeated games with imperfect monitoring. *Games and Economic Behavior 46*(1), 143–173.

Urbano, A. and J. E. Vila (2002). Computational complexity and communication: Coordination in two-player games. *Econometrica 70*(5), 1893–1927.