comprises an efficient solution for this problem. More specifically, we present an efficient algorithm that, for a given function, computes a function that is equivalent to the given function in the sense of protocol equivalence. Assuming the existence of an efficient evaluation function, it will be shown that every protocol can be computed in polynomial time.

**Theorem 1**

Suppose that a function $f$ is given. Then, there exists a protocol $P$ such that $P(f)$ is equivalent to $f$.

**Proof:**

1. **Correctness:** The protocol $P$ always outputs the correct value.
2. **Completeness:** The protocol outputs the correct value for every input.

**Protocol:**

- **Input:** A function $f$.
- **Output:** A protocol $P(f)$.

**Example:**

- Given the function $f(x_0, x_1) = x_0 \land x_1$, the protocol $P(f)$ computes the same output as $f$.

**Extended Abstract**

**How to Solve Any Protocol Problem**

Jerusalem, Israel

Cambridge, MA 02139

MIT

Herta, Israel

Dept. of Computer Sci.

Bar-Ilan Univ.

Any McGregor

Tel-Aviv, Israel

ABSTRACT
Any can obtain $f(x^*)$ except for cases in which a minority of $x$ evaluating $f$ under the condition function.

Indistinguishability is a correct fault-tolerant protocol for computing $f$ if $f$ is a correct fault-tolerant protocol for computing $f$. If the faulty, faulty processors.

For the sake of simplicity, we consider the special case in which the task is to compute one common operation as a function of local inputs. Let $f$ be a function and $N$ be any field. Then $f(x^*)$ is a function of $x^*$.

### 1. Formal Setting

**Protocol:**

- The protocol consists of phases such as a protocol.
- A polynomial-time algorithm can compute a polynomial-time algorithm.
- The complexity of the protocol is polynomial in the size of the function.
- A correct fault-tolerant protocol that computes the maximum degree of $f$ can be computed. The complexity of the protocol is polynomial in the size of the function.
- A correct fault-tolerant protocol that computes the maximum degree of $f$ can be computed. The complexity of the protocol is polynomial in the size of the function.
- A correct fault-tolerant protocol that computes the maximum degree of $f$ can be computed. The complexity of the protocol is polynomial in the size of the function.
- The main result of this paper is an absolute algorithmic reduction of the above problems. If the maximum degree of the function.

### Executing the Protocol

- The protocol requires that only the faulty processors learn about the values of the non-faulty processors. By the protocol, these processors learn about the values of the non-faulty processors.
- The protocol requires that only the faulty processors learn about the values of the non-faulty processors. By the protocol, these processors learn about the values of the non-faulty processors.
- The protocol requires that only the faulty processors learn about the values of the non-faulty processors. By the protocol, these processors learn about the values of the non-faulty processors.
- The protocol requires that only the faulty processors learn about the values of the non-faulty processors. By the protocol, these processors learn about the values of the non-faulty processors.
- The protocol requires that only the faulty processors learn about the values of the non-faulty processors. By the protocol, these processors learn about the values of the non-faulty processors.
and other maximum property were known now.

the function of $f$. To the best of our knowledge, no other non-trivial protocols which are correct
outside our protocol do not exist.

The ideas suggested by Cohen (1970) do not extend to any other function (except multiplication).

other processors is their sum $\sum_{i=1}^{n} x_i^n$, and this of course can be avoided.

the sum of all local inputs $(x_i^n \cdot x^n_i \cdot x^n_i)$ is too large. We therefore look for local inputs or the
participating in the protocol, can be efficiently computed from their local inputs $(1 \cdot x_i^n \cdot f(x))$ and
obtain the maximum $\max$. We have that if a coalition $C$ of players is the set of players which can
compute under $\max$, and the sum function was recently presented by Cohen (1970). In the case of the sum function, we can show by

A correct fail-忍受-protocol which offers the maximum degree of privacy allowed by

the honest players. However, this protocol does not offer any privacy.

the honest players. This allows the inputs of the players which did not participate properly are revealed by
pay protocol this input. The inputs of players who did not participate properly are revealed by
all other players, and no other player (except this player) is the sole participant. For computing the sum function, let each player

Using the notion of verifiable secret sharing (VSS), it is easy to present a correct fail-

The choice has to be made without any knowledge of the local inputs of non-rival players.

A correct fail-忍受-protocol for computing will not allow any players to "show off" the sum

1.2 An Example

the value of $f(x)$.

compute after participating in an execution of $f$. It could have computed from its local inputs and

formity, offers the maximum degree of privacy, irrespective of the local inputs.

or of communicating the outputs of the failure processors after participating in the protocol.

Definition (Extended): We say that the protocol $\mathcal{C}$ offers the maximum degree of privacy allowed by $f$. It offers very good privacy if less than $\frac{1}{2} - \epsilon$ failure processors have ever entered the protocol.

In this context, let the "summed possible one.

non-rival majority will detect the existence of such a minority (and its identity). We believe

is local inputs do not receive any part in the execution. In this latter case, the

Our original construction made use of two such circuits. We assume that the

query-time extension of our protocol is polynomial in the degree of the

function, which is, as we have already shown, at most a polynomial in the

talons of the input.

Given a Turing Machine program and its inputs, we can easily construct a functionally

correct protocol for semi-honest parties.
Now party 1 less be his piece of the current one. Notice that

\[ p \cdot \frac{1}{q} = \frac{1}{q} + \frac{1}{q} \]

The protocol does not leak knowledge about \( q \).

(1)

\[ \frac{1}{q} \]

Party 1 ends with both \( q \) and \( \frac{1}{q} \).

(2)

\[ \frac{1}{q} \]

The new lone is obtained by multiplying two long decimals \( l \) and \( l \).

(3)

The current lone.

in this piece of the current lone. All other parties set their piece of

the case one of the parities (say the first parity) adds the convince to the piece of the lone. Result.

(1)

The new lone is obtained by adding the convince to some previous lone (say lone 1). In this

obtain the value of this lone.

The new lone is an input lone. In this case, the parties already have the pieces allowing to

We distinguish between three cases:

We program sequentially generate pieces for a new lone from the pieces of the previous lone.

The protocol sequentially generates pieces for a new lone from the pieces of the previous lone.

The protocol will only return the value of the last lone in the straightforward program. To end the protocol, will see the sanitizer.

The straightforward program. Our purpose is to allow the parties to hold pieces allowing to obtain the

obtain the values of all the lone in this path. The parties hold pieces allowing to obtain the values of all the lone in this

The corresponding piece of \( q \) is \( \frac{1}{q} \).

Next, the "common" uses the public-key of each parity in rounds to him secretly the

The protocol starts by each party sharing each of his input bits with all other parties using a

here we do not require that \( \alpha < m \) for each.

Note that

then the protocol can be efficiently computed from \( \alpha \), \( \beta \), and \( \gamma \).

Each coalition of a subset of the parities, whenever \( \alpha \) can completely detect when the maximun of inputs allowed by the original function (\( \phi \)) is the best.

Also, for each coalition \( \gamma \) with each of the inputs known to any party, the protocol will order the

we now present a protocol by which we can obtain partners can evaluate a straightforward-p

Through the rest of this section arithmetic will be in \( \mathbb{Z} \).

Such straightforward programs instead use ideas of Shamir. Hence we present here a simpler con-

\[ p, \frac{1}{q} = \frac{1}{q} \]

\[ p \cdot \frac{1}{q} = \frac{1}{q} \]
The protocol deals with a party protocol after the maximum possible privacy is obtained. However, the existence of the protocol can be used as a tool for privacy-preserving communications, especially in multi-party settings where privacy is crucial. In the presence of TRENT, a protocol that satisfies the above conditions exists under more general conditions. We now present a compiler from semi-honest to fault-tolerant protocols for semi-honest protocols.
all games can be played.

The second phase of the combined protocol consists of a "coordinated simulation" of the original protocol, as derived from this extended abstract.

The core idea of the combined protocol preserves the completeness and privacy of the original protocol, while the property of completeness is retained. The security of the protocol in (GDP) is the sender can prove in zero-knowledge that the messages sent in the original protocol are compounded in polynomial-time when given the primitive and random inputs to the sender and all the messages he has received. Using the proof and random inputs of the sender, the prover can prove in polynomial-time that the "coordinated simulation" of the combined protocol is consistent with the original protocol.
The necessary condition is that the game has a unique, non-trivial, fully-determined solution. This is a property of the game's structure, and the existence of such a solution is what makes the game interesting and meaningful. In general, games with a unique solution are easier to analyze and understand, as they do not require any heuristic or ad-hoc decisions. However, games that lack a unique solution can also be interesting, as they may have multiple possible outcomes, depending on the players' strategies. In such games, the concept of Nash equilibrium becomes particularly relevant, as it provides a way to predict the likely outcomes of the game, given the players' best responses to each other's strategies.
We show that the first question can be affirmatively answered by a computational complexity model. Namely, every game in which the players make computable moves is planable.

Does every game have a model in which it is planable, or should we restrict our attention to only a model (physical or mathematical) which makes all games planable?

Or at least:

This is what we perceive looking in game theory: the attention to the notion of plannability.

Some criteria, particularly if some of the players may be cheating.

However, assume we define NEwPOKer as follows: a player may see his own hand and no other.

In this paper, we will consider the interaction of a game with complicated knowledge functions. It is not all easy to decide whether his players cannot implement the knowledge function in a game that can be implemented by the players without invoking any internal parts. In general, games that are not played in a physical way may not be considered as plannable (planar).

Let us first look at the notion of a (transformed) planar game. This is a trivial game, but one in which you may describe another on another

...
ACKNOWLEDGEMENTS

We are very grateful to Shimon Even, Mike Merritt, Dick Karp, Albert Meyer and Yoram Moses for questioning the "completeness" of our preliminary results. In particular, Dick Karp suggested games with incomplete information as (probably) the most general formulation for our results. We are also grateful to Benny Chor and Shafi Goldwasser for helpful discussions concerning the issues of this paper. Finally, we wish to thank Stuart Haber for suggesting a simplification to our original construction.