IST-2002-507932

ECRYPT

European Network of Excellence in Cryptology

Network of Excellence

Information Society Technologies

D.PROVI.7
Summary Report on Rational Cryptographic Protocols

Due date of deliverable: 31. January 2007
Actual submission date: 2. March 2007

Start date of project: 1. February 2004 Duration: 4 years

Lead contractor: University of Aarhus (BRICS)

Revision 1.0

Project co-funded by the European Commission within the 6th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

Summary Report on Rational Cryptographic Protocols

Editors
Jesper Buus Nielsen (BRICS)

Contributors
Joël Alwen, Christian Cachin,
Jesper Buus Nielsen, Olivier Pereira,
Ahmad-Reza Sadeghi, Berry Schomakers,
Abhi Shelat, Ivan Visconti,

2. March 2007
Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Abstract

This report gives an overview of some of the models and techniques in *rational cryptography*, an emerging research area which in particular uses the methodologies and techniques of *game theory* to analyze cryptographic protocols and which uses *cryptographic protocol theory* to implement game theoretic mechanisms.

Contents

Chapter 1

Introduction

This document gives an overview of models, techniques and open problems in the field of *rational cryptography*. This is a rather new research area which is currently emerging, as part of the larger trend of *algorithmic game theory*. This introduction places rational cryptography as part of algorithmic game theory and discusses the current research trends and challenges of rational cryptography.

Computer science concerns itself with the design and analysis of computational systems. In recent years, huge systems consisting of numerous components run by various agents have become the norm rather than the exception, the prime example being the Internet and subsystems thereof. These systems are characterized by the fact that the agents running them may have very different reasons to participate in the system and possibly conflicting goals and interests. The classical correctness and "goodness" criteria of computer science for understanding computational systems are in general inadequate for understanding such systems as these deal with the case where everybody participating legally in the system have common goals. A more sophisticated point of view is given by the rapidly expanding field of *algorithmic game theory*. The main purpose of algorithmic game theory is to provide an understanding of computational systems based on the tools and language of *game theory*, a mathematical theory of economics designed precisely to analyze systems consisting of several agents with partially conflicting interests. For example, traditionally, cryptographic protocols have been analyzed by assuming that some agents were "good", behaving according to the protocol (even if it might hurt their self-interest) while others were "bad", trying to break the protocol for unknown reasons. Typically, a cryptographic protocol was then shown to behave as desired under the assumption that at least some fixed fraction of the agents participating were good. The game-theoretic point of view is that the "good/bad" model of agents should be replaced with the more realistic assumption that all participating agents are neither good nor bad, but simply act so as to maximize their own self-interest.

Algorithmic game theory is a field in huge growth internationally. In addition to being interdisciplinary between economics and computer science, it is also interdisciplinary within computer science itself, involving the communities of algorithms, cryptography, computational complexity theory, and artificial intelligence. We may identify the currently most important and active subfields of algorithmic game theory as the following.

Game-Theoretic Network Design and Network Routing

Today, huge amounts of traffic are being routed online on the Internet and the underlying networks through an ad hoc collection of existing protocols designed for a situation with only a tiny fraction of that traffic. The amounts of traffic will increase even further as all voice traffic is moved to the Internet and streaming video applications gain popularity. Directing the flow of traffic so that congestion is avoided and so that the cost is optimized is a major concern of the telecom industry. The decisions made are to be taken locally by agents which are also competitors. However, classical network optimization algorithms assume global control and a common understanding of what to optimize. Algorithmic game theory gives an alternative understanding of networks and network protocols. Though the volume of this research is already large, the theory is still in its infancy. Examples of research can be found in [KP99, RT00, Rou05, Rou03].

Auction Theory for Digital Goods and Implementations of Auctions

Auction theory is a classical topic in applied game theory. However, auctions for *digital goods*, such as goods that can be copied at no cost has not been studied much. Also, the Internet has caused the structural complexity of the auctions that are to be considered to increase beyond the classical theories. For instance, even today, for each query performed by Google, an auction takes place for the corresponding advertising space. Each advertiser thus participates in a long sequence of auctions that must be analyzed as a whole. Classical Bayesian analysis of such auctions (and much simpler ones) becomes difficult because of the complexity and because of the fact that it is not easy to define the "right" prior distribution on the types of the players. One point of view that has been advocated recently by algorithmic game theory is to replace the Bayesian model by a worst-case analysis. This can also be viewed as a contribution of algorithmic game theory to classical auction theory. Applying these concepts to complex auctions, such as Google's is still a challenging task. Another important issue is the practical implementation of classical auctions concerning physical goods in an online setting, typically involving the cryptographic discipline of multiparty computation. Examples of research can be found in [NR99, FGHK02, GHKS04, AFG$^+$05].

Equilibrium Computation

Game theory suggests that games are to be understood through their equilibria. The games considered by algorithmic game theory are typically larger by orders of magnitude in combinatorial complexity (such as number of states, number of strategies, number of players, etc.) than those considered in the classical game-theoretic literature. The equilibria of the latter were typically found by ad hoc paper-and-pencil analysis; this is out of the questions for the games we now want to consider. Thus, the field of *equilibrium computation* is also gaining in importance: Given a game we want to automatically compute some representation of one or all the equilibria the game possesses. Two very important contributions of algorithm game theory to this topic is the result of Papadimitriou [Pap05], describing the computation of correlated equilibria for a very large class of games and the result of Chen and Deng [CD05] establishing the computational equivalence of a large number of equilibrium computation tasks. The main open problems concern the computation of equilibria in stochastic and recursive games. Somewhat surprisingly, these open problems have strong connections to important open problems in the field of program verification, in particular model

checking [Jur00]. Other examples of research can be found in [PR05, GP06, DGP06, MS06].

Game-Theoretic Cryptography

Cryptography as a field has always concerned itself with multiagent systems. Traditionally the security models divided the agents in *good* and *bad*. Security e.g. meant protecting the privacy of the good agents under arbitrary behavior from the bad agents. This view was motivated by imagining the bad agents as computers being taken over by a hacker and the good agents as the computers still running the original protocol. This is called the *Byzantine framework*. Recently cryptographic protocols have been designed and analyzed under the sole assumption that the parties are rational. The interest in the interplay between game theory and cryptography is emerging as a separate research area, usually called *rational cryptography* in the cryptographic community.

Rational cryptography is drawing on many notions and techniques from game theory. Rational cryptography e.g. considers rational secure multiparty computation, where the models and techniques of traditional multiparty computation are further developed to deal with the case where parties are not good or bad, but all rational. One issue is to design protocols in which the agents have no incentive to behave in other ways, knowing that the other agents have been suggested to follow the same protocol. Mixed models have also been proposed, where some fraction of the parties are assumed to be rational and the rest assumed to collude and behave arbitrarily, and therefore might deviate from the protocol even without an incentive. This mixes the Byzantine and game-theoretic framework. Other issues that have been considered is using incentive and cryptography to make rational parties provide correct inputs to a computation, and using cryptography to implement the trusted mediator in game-theoretic designs.

So far, there have only been a few examples of such game-theoretic design and analysis of cryptographic protocols but these examples have been extremely encouraging. We predict that the impact of the theory to cryptography will be great. The rest of this document will be dedicated to giving an overview of the research in rational cryptography.

Contents

In Chapter 2 we give a brief overview of some of the game-theoretic notions used in subsequent chapters. This chapter is meant for the reader not familiar with this subject. In Chapter 3 we give a brief introduction to the notion of secure function evaluation from cryptographic protocol theory. This chapter is meant for the reader not familiar with this subject. In Chapter 4 we give an overview of some of the new models which have been proposed for game-theoretic modeling and analysis of cryptographic protocols, and we survey some of the results. The chapter focuses on rational secure function evaluation and non-cooperative function evaluation. In Chapter 5 we survey some of the applications of cryptographic protocol theory to problems from economy. This chapter focuses on auctions, fair division, collaborative benchmarking and forecasting, and polling.

Chapter 2

Introduction to Game Theory

2.1 Strategic Form Games and Nash Equilibria

The *strategic form game* is the game-theoretic starting point for modeling an interaction among n players. In such a game, each player i has his own finite set of actions, A_i, to choose from and his own utility function u_i mapping $A_1 \times \ldots \times A_n$ into the real numbers. (All action sets and utility functions are common knowledge of the players.) The game is played in a single stage without any communication. Each player can be thought to be isolated in his own room facing a panel of buttons, one for each of his possible actions. After each player i plays his chosen action, a_i, by pushing the corresponding button, the outcome of the game—i.e., the profile of actions $a = (a_1, \ldots, a_n)$—is defined, and each player i receives his own payoff $u_i(a)$. More formally,

Definition 2.1 ***(Strategic-Form Games)*** *A* game *is a 3-tuple* $G = \langle I, (A_i)_{i \in I}, (u_i)_{i \in I} \rangle$ *where* I *is a set of* players*,* A_i *is a set of* actions *(also called* pure strategies*) for player* $i \in I$*, and* $u_i : A \to \mathbf{R}$ *is a* payoff *(also called* utility*)* function *for player* $i \in I$ *which maps the set of* outcomes*,* $A = \prod_{i \in I} A_i$*, to a real number.*

A game is *non-cooperative* if the players have no external mechanisms to compel a player to play a specific action, and hence, no way to enforce agreements between players. For convenience, we shall use the notation $[n]$ to denote a set of players $\{1, 2, \ldots, n\}$ and write a game as $G = \langle [n], (A_i), (u_i) \rangle$ where the subscript i implicitly indexes over $1, \ldots, n$.

Example 2.1 A 2-player game, *Simple*, is represented in Figure 2.1. Following standard conventions, a 2-player game is described by a payoff matrix $(p_{r,c})$, where $p_{r,c}$ consists of the pair whose ith component is the utility of player i when player 1 plays action r and player 2 plays action c. Thus, in particular, if player 1 plays TOP and player 2 plays CENTER, player 1's utility in *Simple* is 1 and Player 2's utility is 2.

Assuming the rationality of all players, can one predict the chosen outcome of a given game? In absence of external enforcing mechanisms, in *Simple* the (TOP,MIDDLE) does not seem a stable prediction. Even if the two players meet and enter a "gentlemen's agreement" to play TOP and MIDDLE respectively, such an agreement would be meaningless, because as soon as player 1 retreats to his chamber to play his action, he will naturally be drawn to move RIGHT and increase his payoff by one. A similar reasoning shows that no other outcome of *Simple* is "stable."

		Player 2		
		Left	Center	Right
Player 1	Top	(0,0)	(1,2)	(2,1)
	Middle	(2,1)	(0,0)	(1,2)
	Bottom	(1,2)	(2,1)	(0,0)

Figure 2.1: Game *Simple*.

The situation improves dramatically if the players use randomness to decide which actions to play. We say an *individual strategy* for a player i is a probability distribution over his own set of actions A_i. In a seminal paper [Nas51], Nash put forward a general notion of *self-enforcing* strategy vectors.

Definition 2.2 ***(Nash Equilibrium)*** *Let $G = \langle [n], (A_i), (u_i) \rangle$ be a game and $\sigma = (\sigma_1, \ldots, \sigma_n)$ a vector of strategies where σ_i is an individual strategy for player i. We say that σ is a Nash equilibrium for G if for all players $i \in [n]$ and for all strategies $\widehat{\sigma}_i$,*

$$u_i(\sigma_i, \sigma_{-i}) \geq u_i(\widehat{\sigma}_i, \sigma_{-i}).$$

In the above definition, following standard notation, σ_{-i} denotes the vector of strategies in σ for all players except i, and the utility function u_i evaluated on a vector of n strategies (rather than an outcome of n actions) refers to i's *expected* utility, arising from the n underlying distributions. We will also use the following, less standard notation.

Notation. Let σ be a strategy vector for a game $G = ([n], (A_i)_{i \in [n]}, (u_i)_{i \in [n]})$. Then, by $\vec{u}(\sigma)$ we denote the vector of expected utilities for all of the players: that is,

$$\vec{u}(\sigma) = (u_1(\sigma), \ldots, u_n(\sigma)).$$

Informally, therefore, in a Nash equilibrium σ, no player i has an incentive to deviate from his strategy σ_i if all other players do stick to their own strategies.

It is immediately seen that the following is a necessary and sufficient condition for σ to be a Nash equilibrium for G: for all players $i \in [n]$ and for all actions $a_i \in A_i$,

$$u_i(\sigma_i, \sigma_{-i}) \geq u_i(a_i, \sigma_{-i}).$$

Example 2.2 In *Simple*, a Nash equilibrium (indeed, the only Nash equilibrium) is (σ_1, σ_2), where σ_i consists of the uniform distribution over A_i. (See Figure 2.) A trivial calculation shows that the expected utility for each player in this Nash equilibrium is 1.

Discussion of Nash Equilibria. In our setting of single-stage, non-cooperative games, Nash equilibria essentially provide the "best outcomes achievable by the players alone." Indeed, no special cooperation or external mechanism is needed for a player i to choose his to-be-played action according to a specified distribution over A_i: he only needs to flip his own trusted coin a few times. Another positive aspect, is that they are guaranteed to exist for every game [Nas51].

Even in our single-stage setting, however, Nash equilibria appear to suffer from two main drawbacks:

	1/3 Left	1/3 Center	1/3 Right
1/3 Top	(0,0)	(1,2)	(2,1)
1/3 Middle	(2,1)	(0,0)	(1,2)
1/3 Bottom	(1,2)	(2,1)	(0,0)

Figure 2.2: A Nash equilibrium for *Simple*. (The probability of each action is printed next to it.)

1. *They remain quite elusive:* though guaranteed to exist, no *efficient* way is known for computing them.
2. *They may offer only modest payoffs:* a bit subjectively speaking, "the best the players can do by themselves often is not very much."

2.2 Correlated Equilibria

Notice that, because players' actions are chosen independently, a Nash equilibrium induces a *product distribution* over the possible outcomes of the game. If the probability that player i selects action a_i in A_i is p_i, then the probability of outcome $(a_1, \ldots, a_n)$ is $\prod_i p_i$. For instance, the Nash equilibrium of Example 2 causes each outcome of *Simple* to occur with probability 1/9 —see Figure 2.2. As a consequence, "undesirable" outcomes may have positive probability: if in a two-player game, we wish the desirable outcomes (a_1, a_2) and (b_1, b_2) to occur with positive probabilities, then player 1 must select a_1 in A_1 with some probability $p > 0$ and player 2 must select b_2 in A_2 with some probability $q > 0$. This, however, forces the outcome (a_1, b_2) to occur with probability $pq > 0$, irrespective of the utility it offers to the players.

	1/3 Left	1/3 Center	1/3 Right
1/3 Top	(0,0) 1/9	(1,2) 1/9	(2,1) 1/9
1/3 Middle	(2,1) 1/9	(0,0) 1/9	(1,2) 1/9
1/3 Bottom	(1,2) 1/9	(2,1) 1/9	(0,0) 1/9

Figure 2.3: Nash equilibrium with induced probabilities over *Simple*'s outcomes.

2.2.1 The Notion of Correlated Equilibria

In 1974, Aumann [Aum74] put forward another notion of equilibrium, called the *correlated equilibrium*, which induces a general *joint* rather than product distribution over the outcomes of a game. Intuitively, a correlated equilibrium is a probability distribution, E, over $A_1 \times \cdots \times A_n$ satisfying the following property: if an n-tuple is chosen according to E and its ith component is privately given to player i as his "recommended" action, then no single player can improve his expected payoff by replacing his recommended action by a different one.

Definition 2.3 (Correlated Equilibrium) *Let $G = (I, (A_i)_{i\in I}, (u_i)_{i\in I})$ be a game and $A = \prod_i A_i$ the set of its outcomes. A* correlated equilibrium *for G is a probability distribution E over A, such that for all players i, and for all ("replacement") functions $\rho_i : A_i \to A_i$,*

$$(*) \qquad \sum_{a\in A} u_i(a_i, a_{-i}) \Pr_E(a) \geq \sum_{a\in A} u_i(\rho_i(a_i), a_{-i}) \Pr_E(a)$$

where $\Pr_E(a)$ is the probability that E assigns to outcome a.

Informally, therefore, as long as all other players stick to their recommendations, each player is better off sticking to his own recommended action: by switching to *any* other action, even one depending on the original recommendation, he cannot improve his expected utility. For our future goals we find it useful to establish the following notation.

Notation: If E is a correlated equilibrium for a game $G = \langle [n], (A_i), (u_i)\rangle$, then we denote by $u_i(E)$ the expected utility of player i in E, that is

$$u_i(E) = \sum_{a\in A} u_i(a_i, a_{-i}) \Pr_E(a).$$

By $\vec{u}(E)$ we denote the vector of expected utilities for all players: that is,

$$\vec{u}(E) = (u_1(E), \ldots, u_n(E)).$$

Example 2.3 In *Simple*, consider the distribution of Figure 2.4 (i.e., the uniform distribution over all non-diagonal outcomes), and let us argue that it is a correlated equilibrium. Assume without loss of generality that outcome Middle, Left is chosen and so player 1's recommendation is Middle and player 2's recommendation is Left. Then, player 2 has no incentive to deviate. Indeed, based on his recommendation Left and his knowledge of the correlated equilibrium E, player 2 concludes that, with equal chance, the chosen outcome is either (Middle, Left) or (Bottom, Left). By switching to Center, say, player 1's utility decreases by 2 in the first case, and increases by 1 in the latter case— thus his total expected utility decreases by 1/2. By symmetry, it is immediately seen that player 2 has no incentive to change to Right either, and that player 1 similarly has no incentive to deviate altogether.

It is easy to see that each player's payoff for this correlated equilibrium is $4/3$.

	Left	Center	Right
Top	(0,0) 0	(1,2) 1/6	(2,1) 1/6
Middle	(2,1) 1/6	(0,0) 0	(1,2) 1/6
Bottom	(1,2) 1/6	(2,1) 1/6	(0,0) 0

Figure 2.4: A correlated equilibrium for *Simple*. The probability of each outcome is printed in the same cell.

Discussion of Correlated Equilibria. In general, correlated equilibria are more advantageous than their Nash counterparts in two important respects.

1. *They can always be found efficiently:* that is, using linear programming, for any game G, a correlated equilibrium for G can be found in time polynomial in the standard binary representation of G. [1]

2. *They may yield better payoffs than Nash equilibria:* indeed, game *Simple* has only one Nash equilibrium, with expected utility 1 for each player, and a correlated equilibrium with expected utility $4/3$ (an improvement of over 33 percent!)

2.3 Extensive Form Games and their Equilibria

In a single-shot game, all players take their actions independently. In order to model more complicated games in which players take turns choosing actions, we use *extensive form games*. Such games are modeled by a game tree in which each node represents a point in the game at which some player chooses an actions. In such games, a *strategy* for a player specifies an action to be chosen at *every* node in the tree—even at nodes of the tree which are *never* reached by the strategy. This requirement arises from the dual role of a strategy: the strategy specifies how player i chooses actions, but also specifies what player i would do at other nodes of the game so that other players can reason about their own decisions.

An extensive form game can be represented as a single-shot game, and therefore extensive form games too have Nash equilibria. A well-known weakness of the Nash equilibrium concept, however, is that a Nash equilibrium strategy can instruct players to choose actions which are not their best-response at nodes in the game which are never reached by that strategy. To illustrate, consider the following two-player game in which player 1 moves before player 2.

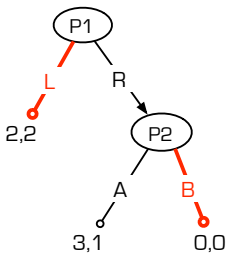

Figure 2.5: An extensive form game in which the Nash equilibrium (L, B) has an "empty threat."

Observe that the strategy profile (L, B) is a Nash equilibrium. The reasoning works as follows: given that player 2 plays B, player 1's best response is to play L, and given that player 1 plays L, player 2's action in node $P2$ does not directly affect the payoffs and so B has the same payoff as A. However, *if* player 2 had to choose an action at that node, choosing B would not be rational given that choosing A has a higher payoff. Thus, player 1 might question whether 2 actually follows through on the "empty threat" of playing B if node $P2$ is reached.

[1] This standard representation is the dense representation of G's matrix that uses at least one bit to represent any matrix entry, even one for which all player utilities are zero. Letting c_i denote the cardinality of A_i, the standard representation of G consists of $c_1 \times c_2 \times \cdots \times c_n$ strings. The string corresponding to outcome $(a_1, \ldots, a_n)$ encodes in binary the corresponding n-tuple of player utilities. Thus, if every player utility is a rational number p/q, and the highest value for such p or q is M, then the number of bits used to represent G is $2n \log(M) \prod_{i=1}^{n} c_i$.

In extensive form games with perfect information, the notion of a *subgame perfect equilibrium* is a meaningful strengthening of Nash equilibria. Unfortunately, games of imperfect information have few, if any, proper subgames. In this case, subgame perfect equilibria are really no more meaningful than Nash equilibria.

In order to capture the desired properties of subgame perfect equilibria in games with imperfect information, Selten introduces the notion of trembling-hand perfect (or simply perfect) equilibrium [Sel65]. Later, Kreps and Wilson [KW82] introduce *sequential equilibria* as a weakening of perfect equilibria. Informally, the strategies of a sequential equilibrium are best responses at *every* history of the extended form game, given some common beliefs about the "state of the game" at each node of the game. Because a sequential equilibrium must be a best-response at every history of the game, such an equilibrium cannot be supported by any "empty threat" such as a minmax punishment. Formal definitions of these concepts follow.

Definition 2.4 (Extensive form game of imperfect information)
An extensive form game of imperfect information *G is a tuple*

$$([n], H, Z, u_i, A_h, P, I_i)$$

where $[n]$ is the (finite) set of players, H is a set of history *sequences which includes the empty sequence, $Z \subset H$ is a subset of* terminal sequences *such that $h \cdot a \notin H$ whenever $h \in Z$, u_i is player i's utility function which maps Z to Q, A_h is a finite set of actions such that the sequence $h \cdot a \in H$ whenever $a \in A_h$, P is a turn function P mapping $H - Z$ to $[n]$, I_i are the information sets which map H to a sequence of strings such that $\forall\, h \cdot a \in H$, $\forall i \in [n]$, $I_i(h)$ is a prefix of $I_i(h \cdot a)$ and $\forall i \in [n]$, $\forall h, h' \in H$, $P(h) = i$ and $I_i(h) = I_i(h')$ implies $P(h') = P(h)$ and $A'_h = A_h$.*

Definition 2.5 ***(Sequential Equilibrium) [KW82]*** *Let G be an extensive form game of imperfect information.*

A belief system *is a set μ, indexed by the information sets $I_h, h \in H$, where μ_{I_h} is a probability distribution over the sequences in I_h.*

Recall that Z is the set of terminal nodes in game G. Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ be a profile of strategies. For any player i and any information set I any sequence $h \in I$, and any terminal history $z \in Z$, let $\Pr[h \xrightarrow{\sigma} z]$ be the probability of eventually reaching z from sequence h when all players use strategy σ. Given a belief system μ, we may now define the conditional probability of reaching a terminal history z when starting from an information set I:

$$\Pr_{\mu}[I \xrightarrow{\sigma} z] = \sum_{h \in I} \mu_I(h) \Pr[h \xrightarrow{\sigma} z]$$

With these probabilities defined, we can now compute the expected value of an information set I under strategy π and beliefs μ as:

$$E_\mu[u_i(\sigma \mid I)] = \sum_{z \in Z} u_i(z) \Pr_{\mu}[I \xrightarrow{\sigma} z]$$

A strategy profile σ is totally mixed *if at every information set, all actions at that information set are assigned nonzero probability. A belief system μ is* consistent *with respect to a strategy profile σ if there exists a sequence of totally mixed strategy profiles $\sigma^1, \sigma^2, \ldots$ such*

that $\lim_{k\to\infty}(\sigma^k, \mu^k) = (\sigma, \mu)$. *Here* μ^k *is uniquely determined by the totally mixed strategy profile* σ^k *via Bayes rule.*

A strategy σ_i' *for* i *is bounded-length with respect to* σ_{-i} *if at every information set, the strategy* σ_i', σ_{-i} *leads to a terminal node in* Z.

An assessment, (σ, μ), *is a* sequential equilibrium *if* μ *is a consistent belief and for every player* i *and every information set* $I \in \mathcal{I}$,

$$E_\mu[u_i(\sigma_i, \sigma_{-i} \mid I)] \geq E_\mu[u_i(\sigma_i', \sigma_{-i} \mid I)]$$

for every bounded-length alternative strategy σ_i' *for* i.

Chapter 3

Introduction to Cryptographic Protocol Theory

In this chapter we give a brief introduction to the notion of secure function evaluation from cryptographic protocol theory. The notion is concerned with what it means for n parties to securely compute a function of their private inputs while leaking no unnecessary information about their inputs.[1] In Section 3.1 we sketch the cryptographic model of secure function evaluation. In Section 3.2 we sketch a general technique which allows to securely compute any functionality in the presence of a so-called honest-but-curious adversary. In Section 3.3 we sketch a general technique which allows to securely compute any functionality in the presence of a malicious adversary.

3.1 Cryptographic Model of Secure Function Evaluation

A functionality $f = (f_1, \ldots, f_n)$ for n parties $P_1, \ldots, P_n$ is a mapping from the inputs $(x_1, \ldots, x_n)$ of the parties to the outputs $(y_1, \ldots, y_n)$, one for each party, computed as $(y_1, \ldots, y_n) = (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n))$. If the privacy of the input is not a concern, the parties can just exchange their inputs and then compute the output without further interaction. If instead each party wishes to keep its input private then things are much more interesting. In an ideal world, in which an on-line trusted third party T is willing to help, parties can compute the functionality f by simply sending their inputs to T who computes the outputs and gives y_i to P_i. Since the requirement of an on-line trusted third party is problematic, it is an important problem to design protocols such that parties can *securely* compute a functionality f without the use of a trusted party. Such protocols are referred to as protocols for *secure function evaluation* (in short, SFE).

Adversarial settings. An *honest-but-curious adversary* corrupts parties that however have to follow the protocol specification (these parties are known as *semi-honest parties*). However, the adversary may attempt to learn more information than she is entitled to by examining the transcript of messages received during the protocol execution. This models e.g. a coalition of parties that pool their view of the protocol to try to learn more than they should.

[1] The output itself will leak information on the inputs, but the protocol should leak no other information.

A *malicious adversary* can, in general, select some parties, control them and potentially decide to not let the selected parties follow the protocol specification. In this case, the popular notation distinguishes *honest parties* from *corrupted parties*, where corrupted parties are the ones that have been selected by the adversary while honest parties have not.

Adversaries can also be partitioned according to when parties are corrupted: the corruption of parties may occur either at the beginning of the protocol (for *static adversaries*) or adaptively during its run (for *adaptive adversaries*). The corrupted parties leak either their current state or both the current state and their history. Security against a static adversary is in general a satisfying security notion under the assumption that secure data erasure is achievable.

Security of the protocols. In cryptography the *simulation paradigm* is the most popular approach to prove the security of a protocol. An *ideal process* is formulated, where a perfectly trusted third party receives the inputs from the parties and sends back to them the corresponding outputs. In the ideal process, the only power of a passive adversary is to see the inputs and outputs of the corrupted parties; A malicious adversary can in addition choose alternative inputs for the corrupted parties. The adversary, however, has no control over the trusted third party. The security of the protocol is then proved by showing that for any real-world adversary attacking the protocol in the real world, there exists an ideal-process adversary attacking the ideal process which obtains the same goals as the real-world adversary, meaning that it collects the same information and makes the process give the same outputs. This shows that the real-world protocol is at least as secure as the ideal process, which is of course the best one can hope for.

A protocol secure against honest-but-curious adversaries is also called a *private protocol* and a protocol secure against malicious adversaries is called a *robust protocol*.

3.2 Yao's Protocol

Yao [Yao86] gave a secure implementation for honest-but-curious adversaries for every two-party efficient functionality. Next we briefly describe Yao's protocol following the presentation of [LP04].

Let f be a polynomial-time functionality (assume for now that it is deterministic) and denote by x_1 and x_2 the inputs of the two parties P_1 and P_2. The first step is to consider the function f as an arithmetic circuit C over $GF(2)$. Circuit $C(x_1, x_2)$ is evaluated gate-by-gate, from the input wires to the output wires. A gate is a function $g : \{0,1\}^2 \to \{0,1\}$. When the input wires to a gate g hold the values $\alpha, \beta \in \{0,1\}$, the outgoing wires of the gate hold the value $g(\alpha, \beta)$. The output of the circuit is the values held by the output wires of the circuit. Thus, essentially, evaluating a circuit involves allocating appropriate zero-one values to the wires of the circuit. In the description below, we refer to four different types of wires in a circuit: circuit-input wires (that receive the input values x_1 and x_2), circuit-output wires (that carry the value $C(x_1, x_2)$), gate-input wires (that enter some gate g), and gate-output wires (that leave some gate g).

In a private protocol, the only value learned by a party P_i should be its output y_i. Therefore, the values that are allocated to all wires that are not circuit-output, should not be learned by either party (these values may reveal information about the other party's input that could not be otherwise learned from the output). Moreover, each party should only get

an appropriate part of the output wires. The basic idea behind Yao's protocol is to provide a method of computing a circuit so that values obtained on all wires other than the corresponding circuit-output wires are never revealed. Therefore, for each gate g, the input values α, β of g are not known by the two parties. Instead P_i, $i = 1, 2$, holds randomly chosen a_i, b_i such that $a_1 + a_2 = \alpha$ and $b_1 + b_2 = \beta$. We call a_i, b_i *shares*. There are two types of gates in C: addition gates and multiplication gates. Addition gates are evaluated by each party locally adding its shares of the input values. Specifically, suppose P_i, $i = 1, 2$, holds a_i, b_i (shares of the input wires α, β of an addition gate). Then $a_i + b_i$ is a share of the output value $\gamma = \alpha + \beta$ of the addition gate. Multiplication gates are instead evaluated using a so-called *1-out-of-4 oblivious transfer* protocol.[2]

Yao's protocol works only for two parties. A more complex approach was developed for the case of $n > 2$ parties by Goldreich, Micali, and Wigderson [GMW87].

3.3 The GMW Protocol

The result of [Yao86, GMW87] solved the SFE problem for honest-but-curious adversaries. We now sketch how to solve the problem for malicious adversaries. In [GMW87], it was shown how to enforce potentially malicious adversaries to behave as semi-honest parties. Specifically, in [GMW87] it is shown how to compile any protocol for a functionality f that is secure for honest-but-curious adversaries into a protocol for the same functionality that is secure for malicious adversaries. The protocol requires that all communication is done by broadcast. Since the private protocols discussed in Section 3.2 have, or can be made to have, this property, the GMW compiler can be applied to e.g. the private protocols discussed in Section 3.2 to obtain a robust protocol for any functionality.

The GMW compiler begins by having each party *commit* to its input.[3] Next, the parties run a coin-tossing protocol in order to fix their random tapes. A simple coin-tossing protocol in which all parties receive the same uniformly distributed string is not sufficient here. This is because the parties' random tapes must remain secret. Instead, an augmented coin-tossing protocol is used, where one party receives a uniformly distributed string (to be used as its random tape) and the other parties receives a commitment to that string. Now, following these two steps, each party holds its own input and uniformly distributed random tape, and a commitment to the other parties' input and random tapes. Next, the commitments to the random tape and to the inputs are used to "enforce" semi-honest behavior. Observe that a protocol specification is a deterministic function of a party's view consisting of its input, random tape and messages received so far. Further observe that each party holds a commitment to the input and random tape of the other parties and that the messages sent so far are public (they have been broadcast). Therefore, the assertion that a new message is computed according to the protocol is an NP statement (and the party sending the message knows

[2] In a 1-out-of-4 oblivious transfer protocol one of the parties, Alice, has four input bits a_1, a_2, a_3, a_4 and the other party, Bob, has a selection $b \in \{1, 2, 3, 4\}$. As a result of the protocol Bob learns a_b, and only a_b, and Alice learns nothing about b.

[3] A party commits to a message m by sending a bit string $C = \text{commit}(m, r)$ to the receiver, where r is a randomizer ensuring that two commitments to the same message cannot be linked. A commitment is opened by sending the *opening* (m, r) to the receiver and letting the receiver check that $C = \text{commit}(m, r)$. A commitment scheme should be *hiding* (it leaks no information on m) and *binding* (it is hard to compute (m, r) and (m', r') such that $m \neq m'$ and $\text{commit}(m, r) = \text{commit}(m', r')$).

an adequate NP-witness to it).[4] This means that the parties can use *zero-knowledge proofs* to show that their steps are indeed run according to the protocol specification.[5] Therefore, in the protocol emulation phase, the parties send messages according to the instructions of the basic protocol, while proving at each step that the messages sent are correct. The key point is that, due to the soundness of the proofs, even a malicious adversary cannot deviate from the protocol specification without being detected. Therefore, the adversary is limited to semi-honest behavior. Furthermore, since the proofs are zero-knowledge, nothing "more" is revealed in the compiled protocol than in the basic protocol. We conclude that the security of the compiled protocol (against malicious adversaries) is directly derived from the security of the basic protocol (against honest-but-curious adversaries). In summary, the GMW compiler has three components: input commitment, coin-tossing and protocol emulation (where the parties prove that their steps are according to the protocol specification).

The compiled protocols can be ensured to have the property that even when a minority of the parties deviate from the protocol will the protocol terminate with the correct output and be secure for the remaining parties. If more than half of the parties deviate from the protocol, it might terminate without computing a result. If a result is computed, it will however always be correct. When more than half of the parties deviate, the protocol might however be *unfair* in the sense that some of the parties deviating from the protocol learn the result while the honest parties do not.

1-out-of-4 oblivious transfer, commitment schemes and augmented coin-tossing protocols can be built from so-called one-way permutations, such as the RSA function. Hence, under the assumption that trapdoor permutations exist, for any efficient functionality f, there exists a protocol implementing f that is secure against malicious adversaries. The result of the protocol is always correct, but if the honest parties are in minority, then the protocol might fail to provide an output to the honest parties, while still leaking the output to the corrupted parties. For a concise description of robust SFE protocols, we refer to the book of Goldreich [Gol04].

[4] An NP witness for a statement is a bit string which allows to efficiently check the validity of the statement. In this case the NP witness consists of the opening of the commitment to the input and the opening of the commitment to the randomness, as given the inputs and the randomness it is possible to run the party on the known inputs to see if the outputs it sent are computed according to the specification of the protocol.

[5] A zero-knowledge proof allows to prove the validity of a statement without leaking any information but the validity of the statement. In particular, no information on the witness is leaked. It is known that every statement which has an NP witness can be proved in zero-knowledge efficiently.

Chapter 4

Game Theory Applied to Cryptographic Protocols

4.1 Introduction

This chapter surveys some of the existing models and techniques in rational cryptography. In Section 4.2 we sketch a framework for expressing games as protocols, and protocols as games. In Section 4.3 we then exemplify how to express some central concepts from game theory in this framework. In Section 4.4 we discuss the notion of secure implementation of a mediator, from a cryptographic protocol theory point-of-view and from a game-theoretic point-of-view. The notion of secure implementation is central in much of the research on rational cryptography. In Section 4.5 we give a discussion of the challenges of game-theoretic secure implementation, which is helpful in appreciating some of the complications of the concrete models and techniques surveyed in later sections. In Section 4.6 we sketch some popular assumptions and techniques used by several works, before we in Sections 4.7 to 4.10 survey some concrete models and techniques. In Section 4.11 we give some pointers for further reading.

4.2 Protocols as Games

We start by unifying the terminologies of game theory and protocol theory. Game theory talks in terms of *agents* having *types* and *strategies* which describe the *actions* they take in the *games* in which they take part, and talks about games having outcomes and the agents having *utilities*. Protocol theory talks in terms of *parties* having *inputs* and *programs* which describe the *messages* they send in the *protocols* in which they take part, and talks about protocols having *outputs* and parties having *outputs*. The correspondence between *utilities* and *outputs* is not completely clear, and the main motivation for developing a unified framework is to clarify this correspondence. The correspondence between the other notions is intuitively much more clear, but it is still convenient to be explicit about the correspondence. We do this by developing one unified formal framework allowing to describe games as protocols and protocols as games.

The terms *agent* and *party* are used informally in the respective frameworks, an agent being used as a term to which we can associate types, strategies, actions and utilities, and a party being used as a term to which we can associate inputs, programs and outputs. We will

primarily use the term *party*.

We will unify the notions of a *strategy* and a *program* using the notion of an *input-output automaton*, or *IOA* [LT87]. We will sometimes denote the IOAs as strategies and sometimes denote them as programs. We unify the notion of a *game* and a *protocol* using an *interactive system* of IOAs. We will exclusively call an interactive system a protocol, reserving the term game for something different. We capture the terms *outcome* and *output* using the term *outcome*, where the outcome of a protocol is computed by the IOAs based on their communication. We then capture the notion of utility by letting it be a function of the outcome of the protocol and the inputs of the IOAs.

When defining the outcome of a protocol from the outputs of the parties we are met with the challenge that we typically want the parties to agree on the outcome of a game, which is difficult when the parties can use any strategy they want. In the Byzantine framework this is handled by requiring that only the honest parties are required to agree on the outcome. This notion does not make sense in the game-theoretic model setting, as there are no honest parties. We therefore handle this problem in a different manner. Specifically we will enforce that the parties agree on the outcome. This is done by letting the outcome be a function of the values which all parties are guaranteed to know. All interaction between the parties in a protocol will take place using communication resources, and we will introduce the notion of common outputs of such resources, the common outputs essentially being the values that the resource is guaranteed to deliver to all parties in the protocol. We then let the common outcome of a protocol be some *fixed* function Ω of the common outputs of the resource. The function Ω is thus part of the specification of the game being played. The strategy of a party then comes down to picking its own local behavior, and cannot change the communication resources or how the outcome is computed from the common outputs of these resources. For generality we also allow the utility of a party to be a function of its own private output. This is necessary when the utility of a party is determined by how much information it can compute about the inputs of the other parties, for instance.

4.2.1 Input-Output Automatons

An *IO* automaton is for our purpose a quadruple $IOA = (\mathrm{In}, \mathrm{Out}, \mathrm{State}, A)$, where In is a finite set of input ports, or *inports*, Out is a finite set of output ports, or *outports*, State is the set of states the automaton can be in, and A is an algorithm describing the behavior of the automaton. Without going into details, an IOA keeps a state $state \in \mathrm{State}$. We then say that an IOA can be *activated* with some inputs x_i on some of its inports $I_i \in \mathrm{In}$. By this we mean that the algorithm A is given the values (I_i, x_i) and the current state $state'$ of the IOA. Then A outputs a new state $state'$ and values of the form (O_i, y_i), where $O_i \in \mathrm{Out}$. We say that IOA changes state to $state'$ and outputs the values y_i on O_i.

When discussing asymptotic efficiency we give some common *security parameter* $\kappa \in \mathbf{N}$ to all IOAs in the system. The behavior of the IOAs might depend on κ, typically by using cryptographic keys of length κ. As usual we say that an IOA IOA is efficient in κ if all activations of IOA can be computed in polynomial time in κ.

4.2.2 Resources

A *resource* is an IOA $\mathcal{R}$ which allows n protocol parties to communicate. It has n inports, named $\mathtt{in}_{\mathcal{R},1}, \ldots, \mathtt{in}_{\mathcal{R},n}$, and n outports, named $\mathtt{out}_{\mathcal{R},1}, \ldots, \mathtt{out}_{\mathcal{R},n}$. This means that when

a resource $\mathcal{R}$ is activated it outputs values $(y_1, \ldots, y_n)$ (y_i on $\mathtt{out}_{\mathcal{R},i}$), where these values are computed from its current state plus any messages arriving on its inports since its last activation. In addition the resource might change state. We assume that each output y_i is of the form $y_i = (y'_i, y)$, where y is common for for $i = 1, \ldots, n$. This means that $(y_1, \ldots, y_n) = ((y'_1, y), \ldots, (y'_n, y))$. We call y the *common output* from $\mathcal{R}$ and we call y'_i the *private output* to P_i.

4.2.3 Games

A *protocol game* or just *game* is a quadruple $\text{Game} = (\mathcal{R}, \Omega, u, X)$, where $\mathcal{R}$ is a resource, Ω is a function specifying the common output of a protocol playing Game from the common outputs of $\mathcal{R}$, $u = (u_1, \ldots, u_n)$ where u_i, the *utility of* P_i, is a real-valued function defining the utility of P_i from the output distribution of the game, and X is some source providing the inputs. The utilities can in general be functions of all outputs of the game, not just the common ones, and can in general be functions of the output *distributions* as opposed to being defined as functions of the outputs. We soon return to how outputs of games and utilities of games are computed.

4.2.4 Protocol Parties

A *protocol party* P_i (for resource $\mathcal{R}$) is an IOA. It has an index i and has an inport $\mathtt{in}_{P,i}$ on which it receives inputs for the protocol and an outport $\mathtt{out}_{P,i}$ on which it delivers outputs from the game. In addition it has an outport $\mathtt{in}_{\mathcal{R},i}$ and an inport $\mathtt{out}_{\mathcal{R},i}$, which it uses to communicate with the resource $\mathcal{R}$. The protocol party P_i has algorithm A_i describing how it computes outputs from inputs. We call A_i the *program of the party* or the *strategy of the party*.

We say that P_i is a *party for the game* $\text{Game} = (\mathcal{R}, \Omega, u, X)$ if the following holds. First of all, P_i is a party for resource $\mathcal{R}$. Furthermore, whenever P_i receives an input y_i on $\mathtt{out}_{\mathcal{R},i}$ (from the resource $\mathcal{R}$), it parses y_i on the form (y'_i, y). Then whenever it outputs a value Y_i on $\mathtt{out}_{P,i}$ in round r, the value is of the form $Y_i = (Y'_i, Y)$, where $Y = \Omega(y^{(1)}, \ldots, y^{(r)})$ and where $y^{(\rho)}$ is the value received from $\mathcal{R}$ on $\mathtt{out}_{\mathcal{R},i}$ as part of (y'_i, y) in round ρ. In other words: when P_i is interacting with $\mathcal{R}$, then Y is computed from the common outputs of $\mathcal{R}$ using the function Ω.[1]

4.2.5 Protocols

An *n-party protocol* π (with name P and resource named R) is specified by n protocol parties $P_1, \ldots, P_n$ (for a protocol named P and using a resource named R) and a resource $\mathcal{R}$ named R. We say that π is a protocol for the game $\text{Game} = (\mathcal{R}, \Omega, u, X)$ if each P_i is a protocol party for the game Game.

Formally, the protocol π specified by $(P_1, \ldots, P_n)$ will be a new IOA. It has inports $\mathtt{in}_{P,1}, \ldots, \mathtt{in}_{P,n}$ and outports $\mathtt{out}_{P,1}, \ldots, \mathtt{out}_{P,n}$. In each activation the protocol can be run on an input $\vec{X} = (X_1, \ldots, X_n)$ (on $(\mathtt{in}_{P,1}, \ldots, \mathtt{in}_{P,n})$) and will deliver an output $\vec{Y} = (Y_1, \ldots, Y_n)$ (on $(\mathtt{out}_{P,1}, \ldots, \mathtt{out}_{P,n})$). The output is computed by letting the protocol parties interact via the resource R. When this interaction is going on we say that the *protocol is active* and we also say that the *parties are playing a game*. Such activations of the protocol can happen

[1] This convention is introduced to *enforce* that all parties agree on the common outcome of the protocol.

several times in sequence, i.e., the parties can play several games in sequence.[2] The details are given in Fig. 4.1.

The protocol π runs in a sequence of *protocol activations*. In activation number a an input $\vec{X}^{(a)} = (X_1^{(a)}, \ldots, X_n^{(a)})$ is provided, and an outcome $\vec{Y}^{(a)} = (Y_1^{(a)}, \ldots, Y_n^{(a)})$ is defined. During each game the protocol parties interact via the resource $\mathcal{R}$. The execution proceeds as follows:

1. Let $a = 1$.
2. *First the inputs for activation number a is provided:*
   (a) Let $\vec{X}^{(a)} = (X_1^{(a)}, \ldots, X_n^{(a)})$ be the next input for the protocol.
   (b) For $i = 1, \ldots, n$, input $X_i^{(a)}$ to P_i on $\mathtt{in}_{P,i}$.
3. *Then a number of resource rounds are run:*
   (a) Let $\rho := 1$.
   (b) For $i = 1, \ldots, n$, activate P_i, let $x_i^{(a,\rho)}$ be the value it outputs on $\mathtt{in}_{\mathcal{R},i}$, and input $x_i^{(a,\rho)}$ on $\mathcal{R}.\mathtt{in}_{\mathcal{R},i}$.
   (c) Activate R to get an output $\vec{y}^{(a,\rho)} = (y_1^{(a,\rho)}, \ldots, y_n^{(a,\rho)})$.
   (d) For $i = 1, \ldots, n$, input $y_i^{(a,\rho)}$ to P_i on $\mathtt{out}_{\mathcal{R},i}$.
   (e) If round ρ is not the last round of activation number a, then let $\rho := \rho + 1$ and go to Step 3b.
4. *Finally the outcome of activation number a is computed:*
   (a) For $i = 1, \ldots, n$, activate P_i and let $Y_i^{(a)}$ be the value output on $\mathtt{out}_{P,i}$.
   (b) The output of the activation is $\vec{Y}^{(a)} = (Y_1^a, \ldots, Y_n^a)$.
   (c) Let $a := a + 1$, and go to Step 2.

Figure 4.1: The execution of a protocol π specified by $(P_1, \ldots, P_n, R)$.

In Fig. 4.1 it is unspecified where the inputs to the protocol π come from. To complete the picture we use X to denote some source which provides the inputs for the protocol, and we then use $\pi(X)$ to denote an execution of π on the inputs provided by X. I.e., first X provides $\vec{X}^{(1)}$ and π is run on $\vec{X}^{(1)}$ to provide an outcome $\vec{Y}^{(1)}$. Then X provides $\vec{X}^{(2)}$ and π is run on $\vec{X}^{(2)}$, and so on. When after some A activations X provides no more inputs, the execution halts. We use $\pi(X) = (\vec{X}^{(1)}, \vec{Y}^{(1)}, \ldots, \vec{X}^{(A)}, \vec{Y}^{(A)})$ to denote the values thus produced, and we call $\pi(X)$ the *outcome of the protocol*.

Recall that if π is a protocol for some game Game $= (\mathcal{R}, \Omega, u, X)$, then each $\vec{Y}^{(a)}$ is of the form $\vec{Y}^{(a)} = ((Y_1^{(a)\prime}, Y^{(a)}), \ldots, (Y_n^{(a)\prime}, Y^{(a)}))$, where $Y^{(a)}$ is computed from the common outputs of $\mathcal{R}$. We call $Y_i^{(a)\prime}$ the *private output of P_i* after activation number a, and we call $Y^{(a)}$ the *common output of the protocol π* after activation number a. Often the utilities of a protocol execution depends only on the inputs and the *common* outputs of the protocol, in which case it is convenient to have the notation $\Omega(\pi(X)) = (\vec{X}^{(1)}, Y^{(1)}, \ldots, \vec{X}^{(A)}, Y^{(A)})$.

Note that because π is an IOA and gives outputs of the form $((Y_1', Y), \ldots, (Y_n', Y))$, the

[2] One could have considered more general notions of executions, e.g. allowing the parties to play several games in parallel. For clarity we have chosen as simple a model as possible.

protocol π is also a resource. Observe that the common outputs of a protocol is computed from the common outputs from the resource $\mathcal{R}$. This definition ensures that all n parties will agree on the common outputs of the protocol. Furthermore, if the common outputs of the resource $\mathcal{R}$ can be observed by third parties, then the common outputs of the protocol can also be observed by these third parties. More generally, if a protocol party can prove to some third party what was the common outputs of the resource $\mathcal{R}$, then the protocol party can also prove to this third party what was the common output of the protocol. So, in addition to π again being a resource, it has the property that the type of visibility of the common output is the same as the type of visibility that the resource $\mathcal{R}$ has.

4.2.6 Utilities, Strategies and some Terminology

Consider any game Game $= (\mathcal{R}, \Omega, u, X)$ and any protocol π for Game. We consider $\pi(X)$ as playing the game Game. The utility of P_i in $\pi(X)$ is defined via u_i, as a function of $\pi(X) = (\vec{X}^{(1)}, \vec{Y}^{(1)}, \ldots, \vec{X}^{(A)}, \vec{Y}^{(A)})$. I.e., the utility can depend on the inputs (types) and the outputs of the parties, including the private outputs.

Note that we allow the utility to be a function of the *random variable* $\pi(X)$, which is non-standard in game theory, where utilities are typically associated to outcomes. To see the difference, consider a specific game which has two (common) outputs A and B. Player P_1 gets one dollar if the outcome is A and gets nothing if the outcome is B. Assume that the probability of output A is $p \in [0, 1]$ and that the probability of output B is $(1 - p)$; We write this as $[p : A] + [(1 - p) : B]$. Considering a normal utility function, associating utilities to the outcomes, one could e.g. use the utility function $u_1(A) = 1$ and $u_1(B) = 0$ for P_1. This would give an expected utility of $u_1([p : A] + [(1 - p) : B]) = p$. We consider a more general notion where we allow the utility to be a function of the output *distribution*. In the simple case $[p : A] + [(1 - p) : B]$, this just means that u_i is allowed to be a function of p. This of course still allows to define $u_1([p : A] + [(1 - p) : B]) = p$, but would also allow to let $u_1([p : A] + [(1 - p) : B]) = p$ for $p < 1$ and let $u_1([1 : A] + [0 : B]) = 2$, to model that P_1 gets extra pleasure from knowing that it will always win. Allowing the utility to be a function of the output distribution is rather non-standard, and we will therefore be explicit about it each time we use an utility of this type. So, unless said explicitly, we will use a standard utility, where the utility of a distribution $D = \sum_X [p_X : X]$ is given by $u(D) = \sum_X p_X u(X)$.

4.2.7 Strategies and some Terminology

The *strategy of a party* P_i for game Game $= (\mathcal{R}, \Omega, u, X)$ is defined to be its code A_i. So, the strategy of a party describes how it communicates with the resource.

For a protocol game Game $= (\mathcal{R}, \Omega, u, X)$ we call $(\mathcal{R}, \Omega)$ the *game structure* and we call (u, X) the *game parameters*, as it is typically fixed what resource the parties have access to and how the common output (outcome) of the game is computed after the interaction, whereas the utilities u and the input distribution X might vary. Note that it will typically be the case that the best strategy of a party depends both on the game structure and the game parameters. In particular, the best strategy often depends on the input distribution.

4.2.8 Equilibria and Dominating Protocols

Having defined games, strategies and utilities, we can then adopt the notions from game theory into our protocol framework. As an example we define the notions of equilibrium,

domination and *iterated elimination of dominating strategies* (*IEDS*).

Definition 4.1 *Let* Game $= (\mathcal{R}, \Omega, u, X)$ *be any game, and let* $\pi = (A_1, \ldots, A_n)$ *be a protocol for* Game. *We say that* π *is a* Nash equilibrium *for* Game *if it holds for all parties* P_i *that* $u_i(\pi(X)) \geq u_i(\pi'(X))$ *for all protocols* $\pi' = (\hat{A}_i, \pi_{-i})$.[3]

Definition 4.2 *Let* Game $= (\mathcal{R}, \Omega, u, X)$ *be any game, let* A_i *and* A'_i *be strategies for* P_i, *let* π_{-i} *be a protocol for* Game *with* P_i*'s strategy open, and let* $\pi = (A_i, \pi_{-i})$ *and* $\pi' = (A'_i, \pi_{-i})$. *We say that* A_i weakly dominates A'_i *if*

- $\forall \pi_{-i} \, (u_i(\pi(X)) \geq u_i(\pi'(X)))$ *and*
- $\exists \pi_{-i} \, (u_i(\pi(X)) > u_i(\pi'(X)))$.

I.e., (for P_i*) using* A_i *is never worse than using* A'_i *and it is sometimes strictly better. We therefore write* $A_i > A'_i$. *We say that* A_i *is* weakly dominating *if it weakly dominates all other* A'_i, *and we say that* π *is* weakly dominating *if each* A_i *in* π *is weakly dominating.*

A game can have many Nash equilibria. When choosing between them game theory prefers those which are dominating strategies. When it is not possible to choose a dominating strategy other refinements of Nash equilibria are considered, like the IEDS survivor, which is a strategy which is at least not dominated itself.

We define the notion of an *IEDS survivor* (*IEDSS*) informally. For a party P_i we let $\mathcal{P}_i^{(1)}$ denote the set of all strategies (programs) for P_i, and we let $\mathcal{A}_i^{(1)} = \{A_i\}$ be the subset of strategies $A_i \in \mathcal{P}_i^{(1)}$ which are weakly dominated by some $A'_i \in \mathcal{P}_i^{(1)}$. No rational party will play a strategy $A_i \in \mathcal{A}_i^{(1)}$ as it is sometimes an advantage to play the $A'_i > A_i$ and never a disadvantage. This conceptually creates a new smaller game where each P_i chooses only between the strategies $A_i \in \mathcal{P}_i^{(2)} = \mathcal{P}_i^{(1)} \setminus \mathcal{A}_i^{(1)}$. In this game there might be new weakly dominated strategies[4] This gives rise to a new set $\mathcal{A}_i^{(2)}$ and a new set $\mathcal{P}_i^{(3)}$. Iterating the construction gives rise to a sequence $\mathcal{P}_i^{(l)}$ for $l = 1, \ldots, \infty$. We let $IEDS_i = \cap_{i=1}^{\infty} \mathcal{P}_i^{(l)}$ and call $IEDS_i$ the strategies for P_i which *survive iterated elimination of dominated strategies*. And, we say that $\pi = (A_1, \ldots, A_n)$ is an iterated elimination of dominated strategies survivor (*IEDSS*) if each A_i survives iterated elimination of dominated strategies. A common assumption from game theory is that rational parties will play an IEDSS.

It is easy to see that if $\pi = (A_1, \ldots, A_n)$ is a dominating strategy, then each A_i is an IEDSS, as no dominating strategy can be weakly dominated, and it is easy to construct examples of IEDSS's which are not dominating. The notion of IEDSS is thus a strict weakening of the notion of a dominating strategy. The notion of an IEDSS is central to some of the concrete models that we survey later in the chapter.

[3] Adopting notation from game theory, we use for a joint strategy $\pi = (A_1, \ldots, A_{i-1}, A_i, A_{i+1}, \ldots, A_n)$ the notation π_{-i} to denote $(A_1, \ldots, A_{i-1}, \cdot, A_{i+1}, \ldots, A_n)$ where the strategy of P_i is open, and we let $(\hat{A}_i, \pi_{-i}) = (A_1, \ldots, A_{i-1}, \hat{A}_i, A_{i+1}, \ldots, A_n)$.

[4] Defined as in Definition 4.2 except that $A_i, A'_i \in \mathcal{P}^{(2)}$ and π_{-i} quantifies only over the protocols where $A_j \in \mathcal{P}^{(2)}$ for all $P_j \neq P_i$. I.e., assuming that all P_j stick to strategies from $\mathcal{P}_j^{(2)}$, using A_i is never worse than using A'_i and is sometimes strictly better.

4.2.9 Computational Variants

We can also consider computational variants of the above notions. We start by phrasing the notion of a computational Nash equilibrium. Here, as is usual in cryptography, one only considers efficient protocols. This is necessary to allow the use of computational cryptographic primitives like encryption schemes and digital signatures. As opposed to the usual notion of a Nash equilibrium, a computational Nash equilibrium cannot be defined to give a maximal utility. If e.g. one of the other parties broadcast an encryption of a value which P_i would get some utility out of knowing, then P_i could always use a little more time trying to break the encryption scheme, which would always increase its utility by some positive, yet negligible, amount. To allow this we require that a computational Nash equilibrium could not be bettered more than *negligibly* by using another efficient strategy. As is usual in cryptography we model an efficient protocol by a probabilistic polynomial time (*PPT*) protocol. When considering computational Nash equilibria, also the game itself should be efficient, including $\mathcal{R}$, computing Ω and u and sampling X.

Definition 4.3 *Let* Game $= (\mathcal{R}, \Omega, u, X)$ *be any PPT game, and let* $\pi = (A_1, \ldots, A_n)$ *be a PPT protocol for* Game*. We say that* π *is a* computational Nash equilibrium *for* Game *if it holds for all parties* P_i *that* $u_i(\pi'(X)) - u_i(\pi(X)) \in \text{NEGL}$ *for all PPT protocols* $\pi' = (\hat{A}_i, \pi_{-i})$*, where* NEGL *denotes the set of functions which are at most negligibly larger than zero (asymptotically in* κ*).*[5] *I.e., using another strategy can only increase utility by a negligible amount.*

Definition 4.4 *Let* Game $= (\mathcal{R}, \Omega, u, X)$ *be any PPT game, let* A_i *and* A'_i *be PPT strategies for* P_i*, let* π_{-i} *be a PPT protocol for* Game *with* P_i*'s strategy open, and let* $\pi = (A_i, \pi_{-i})$ *and* $\pi' = (A'_i, \pi_{-i})$ *We say that* A_i computational dominates A'_i *if*

- $\forall \pi_{-i} \, (u_i(\pi'(X)) - u_i(\pi(X)) \in \text{NEGL})$ *and*
- $\exists \pi_{-i} \, (u_i(\pi(X)) - u_i(\pi'(X)) \notin \text{NEGL})$.

I.e., (for P_i*) using* A_i *is never more than negligibly worse than using* A'_i *and it is sometimes non-negligibly better.*

4.3 Some Resources and Types of Games

Before surveying existing work on rational cryptography, it is convenient to take a look at some of the network resources that are considered, and look at how some common types of games are expressed in our framework.

4.3.1 Point-to-Point Communication

The simplest resource is probably secure point-to-point communication. In each round the resource PtP parses the input m_i on $\texttt{in}_{PtP,i}$ as a vector $m_{i,1}, \ldots, m_{i,n}$, where $m_{i,j}$ is thought of as the message that P_i wants to send to P_j. The resource then outputs $(m_{1,j}, \ldots, m_{n,j})$ on $\texttt{out}_{PtP,j}$.

[5] Formally, a function $\epsilon : \mathbf{N} \to \mathbf{R}$ is in NEGL if for all constants $c \in \mathbf{N}$ there exists κ_c such that $\epsilon(\kappa) < \kappa^{-c}$ for all $\kappa \geq \kappa_c$.

4.3.2 Broadcast

The notion of a *broadcast channel* plays an important role in many protocols in protocol theory. A broadcast channel allows a sender to broadcast a message in such a way that it is guaranteed that all other parties receive the same message m, even if the sender, by actively deviating from the protocol, tries to create a situation where different parties receive different values. In fact, even if the sender colludes with a subset of the receivers, it should be guaranteed that the remaining receivers receive the same message.

There are two main types of broadcast channels considered in cryptographic protocol theory. A *simultaneous broadcast channel* means that it is possible for the parties to each broadcast a message, in parallel, in such a way that no party can choose its message to be broadcast based on the messages of the other parties. This is opposed to a *rushing broadcast channel* where parties that so desire (the corrupted parties in the Byzantine framework) can choose to see the message broadcast by other parties before sending their own.

For our purpose we define the *rushing broadcast resource RBC* as follows. On input x_i on $\mathtt{in}_{BC,i}$ it outputs the private message $\perp$ and the common message x_i to all other parties. We define the *simultaneous broadcast resource BC* as follows. On input x_i on $\mathtt{in}_{BC,i}$ it outputs the private message $\perp$ and the common message $\perp$ to all other parties. In the next round it then outputs the common message $(x_1, \ldots, x_n)$ to all parties.

4.3.3 Strategic Form Games

We describe how the notion of a *strategic form game* can be captured in our framework. Recall that in a strategic form game each party is given a type t_i and performs one action a_i, independent of the other parties, and then the utility of each P_i is defined from the actions $(a_1, \ldots, a_n)$ and the types $(t_1, \ldots, t_n)$, using its utility function u_i.

We can e.g. capture this notion using a game with the simultaneous broadcast resource, SBC: We let each P_i have input $x_i = t_i$ on $\mathtt{in}_{P,i}$, we let the common output $\Omega = (m_1, \ldots, m_n)$ be the messages broadcast in the first round, and we let $u_i = u_i((x_1, \ldots, x_n), \Omega)$. It would also be possible to formalize it using a game with no resource, where P_i has input $x_i = t_i$, where the output of the game is considered to be the messages $(y_1, \ldots, y_n)$ output by the parties in the first round, and where $u_i = u_i((x_1, \ldots, x_n), (y_1, \ldots, y_n))$. Which model makes best sense depends on the setting modeled. We will typically use the second model, where the actions are private outputs, as the action performed by a party might not be visible to the other parties.

4.3.4 Mediated Games

We can also capture the notion of a *mediated game*. In a mediated game the parties have a type t_i, and first talk to some mediator M. Then each P_i performs some single action a_i, independent of the other parties, and the utility of a party P_i is defined from the actions $(a_1, \ldots, a_n)$ and the types $(t_1, \ldots, t_n)$, using its utility function u_i.

We can capture this notion with a game where P_i has input $x_i = t_i$ on $\mathtt{in}_{P,i}$ and where first the parties have access to the resource M, and then in the first output round outputs some a_i on $\mathtt{out}_{P,i}$, and where the utility of P_i is computed as $u_i(\pi(X)) = u_i((t_1, \ldots, t_n), (a_1, \ldots, a_n))$.

Sometimes mediated games are formulated in a slightly different manner, letting the utilities being a function of a common output of the game, determined by M. We can capture this notion with a game where P_i has input $x_i = t_i$ on $\mathtt{in}_{P,i}$ and where first the parties have

access to the resource M until at some point M outputs a common value y. Then the utilities are computed as $u_i = u_i((t_1, \ldots, t_n), y)$. With this type of mediation the parties have no room to "cheat" after the mediation.

To distinguish the two types of mediated games we will call the latter a *strongly mediated game* and we will call the former a *weakly mediated game*.

It is natural to ask why we even have a notion of a weakly mediated game. Why do we not always let the mediator announce the outcome of the game? The reason is that the final actions $(a_1, \ldots, a_n)$ might be of a form which the mediator cannot perform. Maybe the parties are playing a game to coordinate which items each party bids on in a later auctions. Here the mediator could e.g. be a computer to which the parties input their valuation of the items and where the computer then outputs secretly to each party P_i which items P_i should bid on. In this case the mediating computer can only give the parties recommendations, but cannot perform the actual bidding for the parties.

4.3.5 Dominating Penalized Abort

Sometimes games allow the parties to abort during the game, by refusing to perform some action needed to compute the true outcome of a game. Any complete game has to specify what happens in the case of a party aborting, and what are the utilities in this case. This gives rise to a notion which we will call *dominating penalized abort*. We discuss the notion in the context of strongly mediated games. Consider any strongly mediated game:

- First each P_i, having input x_i, inputs some X_i to the mediator M.
- Then M outputs some common output y to all parties.
- The outcome of the game is $\Omega = y$.
- The utility of P_i is $u_i((x_1, \ldots, x_n), \Omega)$.

We can modify this game into a strongly mediated game with abort, as follows.

- First each P_i, having input x_i, inputs some X_i to the mediator M.
- Then M outputs some common output y to all parties.
- Then each party P_i broadcast a bit $b_i \in \{0, 1\}$, where $b_i = 0$ indicates that P_i wants to abort the game.
- When $b_i = 1$ for all P_i, then the outcome of the game is $\Omega = (1, y)$. If $b_i = 0$ for some P_i, then the outcome is $\Omega = (0, b_1, \ldots, b_n)$.
- The utility of P_i is $u'_i((x_1, \ldots, x_n), \Omega)$.

The natural definition of u' is to let $u'_i((x_1, \ldots, x_n), (1, y)) = u_i((x_1, \ldots, x_n), y)$. The definition of the utilities $u'_i(x_1, \ldots, x_n, , (0, b_1, \ldots, b_n))$ is called the extension of u'. It is clear that this strongly mediated game with abort can have equilibria very different from the equilibria of the strongly mediated game (without abort). It is, however, also intuitively clear that by extending u' properly, it is possible to enforce that the strongly mediated game with abort has *exactly* the same equilibria as the strongly mediated game. One approach is to define $u'_i((x_1, \ldots, x_n), (0, b_1, \ldots, b_n)) = -p$ for some large penalty $p \in \mathbf{R}$ when $b_i = 0$, so that

it is never rational for P_i to abort.[6] When the extension of u' is fixed such that the two games have the same equilibria (of the same strength), we say that the game that allows aborting has *dominating penalized abort*.

4.4 Secure Implementation

Cryptographic protocol theory talks in terms of *protocols implementing resources*. Game theory talks in terms of *games implementing mechanisms* and *strategies implementing special equilibria*. I.e., in our terms, game theory talks in terms of *games implementing resources* and *protocols implementing special equilibria*. Note that, in particular, game theory does *not* talk about protocols implementing resources. A protocol can implement an equilibrium, not a resource, and a resource can be implemented by a game, not a protocol. This might lead to some confusion, at least of terms, when coming from cryptographic protocol theory, so we will address all three notions and compare them.

4.4.1 Protocols Implementing Resources (Cryptographic Protocol Theory)

A central notion, or maybe *the* central notion, in cryptographic protocol theory is that of a protocol being a *cryptographic secure implementation* of a resource.

The typical setting is that some, idealized, resource $\mathcal{R}$ is given, which is on one hand a very powerful tool for building other protocol, but which on the other hand does not exist as a physical resource. Then another, real, resource $\mathcal{S}$ is introduced where on one hand $\mathcal{S}$ is a model of an existing physical resource, but which on the other hand is not nearly as powerful as $\mathcal{R}$. Then it is asked whether the idealized resource $\mathcal{R}$ can be implemented given the real resource $\mathcal{S}$. As an example, it could be that $\mathcal{R} = SBC$ is the resource for simultaneous broadcast (which does not exists physically on e.g. the Internet) and that $\mathcal{S}$ is a model of authenticated point-to-point communication between the parties (which can be implemented on the Internet). One would in this case be asking whether authenticated point-to-point communication allows to securely implement simultaneous broadcast.

To *cryptographic secure implement* $\mathcal{R}$ using $\mathcal{S}$ means giving a protocol $\gamma[\mathcal{S}]$ which uses the resource $\mathcal{S}$, such that $\gamma[\mathcal{S}]$ is as secure as $\mathcal{R}$. By $\gamma[\mathcal{S}]$ being as secure as $\mathcal{R}$ is then meant that if one takes any other protocol $\pi[\mathcal{R}]$ which uses the resource $\mathcal{R}$ and which performs some task T, then the protocol $\pi[\gamma[\mathcal{S}]]$ will also perform the task T. Here $\pi[\gamma[\mathcal{S}]]$ is the protocol $\pi[\mathcal{R}]$ where all activations of the resource $\mathcal{R}$ are replaced by activations of the protocol $\gamma[\mathcal{S}]$. This essentially means that in any setting where it is secure to use $\mathcal{R}$, it is also secure to use $\gamma[\mathcal{S}]$. This notion allows to give real implementations of ideal resources. A secure implementation γ of simultaneous broadcast using authenticated point-to-point channels would e.g. allow to take *any* secure protocol π designed for the simultaneous broadcast model and deploy it on the Internet by replacing the use of simultaneous broadcast by the use of the protocol γ.

This motivates why securely implementing is defined as general as it is, requiring γ to be secure when used in any extended protocol. We call this type of definition an *extensive definition of cryptographic secure implementation*. It is possible to give relaxed extensive definitions by considering a *context*, which is just a set of protocols π. One then requires that $\pi[\gamma]$ is as secure as $\pi[\mathcal{R}]$ for all π from the context. Even for very restricted contexts,

[6] If coalitions and joint outcomes are considered, p must be large enough that it is never rational for any coalition to let just one of the parties in the coalition abort the game.

an extensive definition is however not very useful when actually attempting to prove secure a given protocol γ, as the proof has to consider all possible protocols π from the context.

A lot of research in cryptographic protocol theory has then been focusing on giving alternative definitions of *cryptographic secure implementation*, where the ideal has been to find a definition with the following two properties:

1. The protocol γ implementing $\mathcal{R}$ is expressed as a simple condition on γ and $\mathcal{R}$ alone.
2. If γ is deemed a secure implementation of $\mathcal{R}$ by this simple condition, then γ also securely implements $\mathcal{R}$ in the extensive sense. I.e., it is secure to use γ in place of $\mathcal{R}$ in any protocol π (from the context).

Property 1 is motivated by the hope that a simple condition would make the work involved in proving security much less. Property 2 is motivated by the fact that having proved security using the simple definition of course should imply that the protocol actually is secure when deployed (in the context). We call such an ideal definition of cryptographic secure implementation an *intensive definition of cryptographic-secure implementation*.

The interested reader can find much more on extensive and intensive definitions of cryptographic secure implementation in [Can01]. The central observation about the cryptographic notions is however that they have exclusively been defined in terms of honest parties versus corrupted parties. I.e., γ has been required to be as secure as $\mathcal{R}$ when running with the same set of corrupted parties. The security has then been measured in how many corrupted parties γ can tolerate and still be as secure as $\mathcal{R}$ (when $\mathcal{R}$ is running with the same set of corrupted parties). This notion makes perfect sense in some settings,[7] but there are many settings where it is hard to justify that some parties in the protocol should blindly follow the suggested protocol, as opposed to deviating with the purpose of increasing its own utility.

4.4.2 Games Implementing Resources (Game Theory)

The motivation in much of the research in rational cryptography has been the observation that in some settings the Byzantine framework of honest parties versus corrupted parties makes little sense, and that game theory with its notion of rationality seems to provide a more suitable framework for analyzing protocols where the protocol parties have different utilities associated to the outcome of the protocol.

The goal in rational cryptography has therefore in some cases been to provide notions of *game-theoretic secure implementation*, equivalent to the notions of cryptographic secure implementation, but phrased in terms of rationality instead of honesty versus corruption.

This research has been confronted with the same problem as the research in cryptographic protocol theory, namely, that it is relatively easy to give an *extensive definition of game-theoretic secure implementation* and that this definition lends itself very poorly to actually proving game-theoretic security (even less than does the cryptographic protocol theory

[7] Consider e.g. a setting where the parties running the protocol have no or little utility associated to the outcome of the protocol except its being correct and its own inputs being kept secret. Maybe the inputs to the protocol are supplied by some third parties and the protocol parties simply run a service computing on these encrypted inputs, for some price payed by the third parties. In this setting it is in the interest of all protocol parties simply to supply correct outputs, to keep up the reputation of the service they provide. In this setting, a cryptographic secure implementation would be an equilibrium. On the other hand, some of the protocol parties might become infected by a computer virus which makes them deviate arbitrarily from the protocol. It is therefore in the interest of all protocol parties to run a protocol which can tolerate as many infected (corrupted) parties as possible.

one). Therefore one of the main goals in rational cryptography has been to give an *intensive definition of game-theoretic secure implementation.* As we will see, several intensive definitions have been proposed, for various types of problems. We will later describe the proposed intensive definitions and discuss how they relate to the natural extensive definitions for the considered tasks. As we will see, many of the intensive definitions which have been put forth have proposed to impose privacy requirements on a protocol to call it a game-theoretic secure implementation. Before discussing the specific models that have been proposed it is therefore instructive to give a simple extensive definition of game-theoretic secure implementation along with a simple example why privacy can be an essential part of an intensive definition.

Replacing Resources by Protocols

We first formalize how resources are replaced by protocols.

Let $\pi[\mathcal{R}]$ be any protocol using a resource $\mathcal{R}$, and let Ω^R be a function which defines the common output of $\pi[\mathcal{R}]$ from the common outputs Y^R of the resource $\mathcal{R}$ in $\pi[\mathcal{R}]$. Let $\gamma[\mathcal{S}]$ be any protocol using a resource $\mathcal{S}$, and let Ω^S be a function which defines the common output of $\gamma[\mathcal{R}]$ from the common outputs Y^S of $\mathcal{S}$ in $\gamma[\mathcal{S}]$.

We then define a protocol $\pi[\gamma[\mathcal{S}]]$ for the resource $\mathcal{S}$. The behavior of the protocol party P_i for $\pi[\gamma[\mathcal{S}]]$ is given in Fig. 4.2.

The protocol party $P_i(P_i^\pi, P_i^\gamma)$ runs as part of its code a copy P_i^π of the corresponding protocol party from π and a copy P_i^γ of the corresponding protocol party from γ. The execution proceeds as follows:

1. In an input round of π, party P_i receives an input $X_i^{(a)}$ which it simply inputs to P_i^π on $\mathtt{in}_{P,i}$.
2. In a resource round of π, the party P_i proceeds as follows:
   (a) First P_i activates P_i^π to receive an output $x_i^{(a,\rho)}$ on the outport $\mathtt{out}_{\mathcal{R},i}$ of P_i^π. *This is a message that was supposed to be sent to* $\mathcal{R}$.
   (b) Instead P_i inputs $x_i^{(a,\rho)}$ to P_i^γ on $\mathtt{in}_{R,i}$, as if P_i^γ had received $x_i^{(a,\rho)}$ in an input round of γ.
   (c) Then P_i activates P_i^γ to receive an output on the outport $\mathtt{in}_{S,i}$ of P_i^γ, and then P_i sends this message to $\mathcal{S}$ on $\mathcal{S}.\mathtt{in}_{S,i}$.
   (d) Then P_i continues running P_i^γ with the resource $\mathcal{S}$. I.e., when $\mathcal{S}$ sends a message to P_i (over $\mathtt{out}_{S,i}$), then P_i inputs it to P_i^γ and activates P_i^γ, and when P_i^γ outputs a message to $\mathcal{S}$, then P_i sends the message to $\mathcal{S}$ (over $\mathtt{in}_{S,i}$). This is continued until an output round of γ is reached.
   (e) When an output round of γ is reached P_i^γ delivers some output $y_i^{(a,\rho)}$ on $\mathtt{out}_{\mathcal{R},i}$. Then P_i inputs $y_i^{(a,\rho)}$ to P_i^π on $\mathtt{out}_{\mathcal{R},i}$ as if $y_i^{(a,\rho)}$ had been delivered by $\mathcal{R}$.
3. In an output round of π, party P_i simply activates P_i^π to receive some output $Y_i^{(a)}$ on the outport $\mathtt{out}_{P,i}$ of P_i^π. Then P_i outputs $Y_i^{(a)}$ on its own outport $\mathtt{out}_{P,i}$.

Figure 4.2: The protocol party $P_i(P_i^\pi, P_i^\gamma)$.

In $\pi[\gamma[\mathcal{S}]]$ we simply replace the use of $\mathcal{R}$ with the use of $\gamma[\mathcal{S}]$. To deem $\gamma[\mathcal{S}]$ a secure implementation of $\mathcal{R}$ we then simply ask that $\pi[\gamma[\mathcal{S}]]$ and $\pi[\mathcal{R}]$ have similar properties for all

protocols π.

For this purpose, consider a game $\text{Game}^R = (\mathcal{R}, \Omega^R, u, X)$ and assume that $\pi[\mathcal{R}]$ is a protocol for this game. We can then in some sense consider $\pi[\gamma[\mathcal{S}]]$ a protocol for $\text{Game} = (\mathcal{R}, \Omega^R, u, X)$ too, as it gives outputs of the same form as $\pi[\mathcal{R}]$. The main technical problem is that in $\pi[\gamma[\mathcal{S}]]$ the common outputs are computed from the common outputs from $\mathcal{S}$. If however $\gamma[\mathcal{S}]$ is a protocol for some game structure $\text{Game}^S = (\mathcal{S}, \Omega^S)$, then $\gamma[\mathcal{S}]$ computes its common outputs Y^R from the common outputs Y^S of $\mathcal{S}$ using Ω^S. Therefore $\pi[\gamma[\mathcal{S}]]$ computes its common outputs by first computing common outputs Y^R of $\gamma[\mathcal{S}]$ from the common outputs Y^S of $\mathcal{S}$, using Ω^S, and then from Y^R computes common outputs Y of $\pi[\gamma[\mathcal{S}]]$, using Ω^R. Therefore $\pi[\gamma[\mathcal{S}]]$ is a protocol for the game $\text{Game} = (\mathcal{S}, \Omega', u, X)$, where $\Omega' = \Omega^R \circ \Omega^S$ denotes the above procedure for computing common outputs Y of a protocol from the common outputs Y^S of $\mathcal{S}$. We can then ask whether $\pi[\gamma[\mathcal{S}]]$ in $(\mathcal{S}, \Omega', u, X)$ behaves as $\pi[\mathcal{R}]$ in $(\mathcal{S}, \Omega^R, u, X)$. More generally, we can ask whether $\text{Game} = (\mathcal{R}, \Omega, u, X)$ and $\text{Game}' = (\mathcal{S}, \Omega^R, u, X)$ allow the same types of equilibria. If this is the case, then we say that the game structure $\text{Game}^S = (\mathcal{S}, \Omega^S)$ game-theoretic securely implements the resource $\mathcal{R}$ (using the resource $\mathcal{S}$).

Definition 4.5 *Let $\mathcal{R}$ be any resource, and let* $\text{Game} = (\mathcal{S}, \Omega^S)$ *be any game structure. We say that* Game *is a* game-theoretic secure implementation *of $\mathcal{R}$ if it holds for all games $(\mathcal{R}, \Omega^R, u, X)$ that $(\mathcal{R}, \Omega^R, u, X)$ and $(\mathcal{S}, \Omega^R \circ \Omega^S, u, X)$ allow the same equilibria; I.e., if for all equilibria $\pi[\mathcal{R}]$ for the game $(\mathcal{R}, \Omega^R, u, X)$ there exists an equilibrium $\pi'[\mathcal{S}]$ for the game $(\mathcal{S}, \Omega^R \circ \Omega^S, u, X)$ such that $u(\pi[\mathcal{R}]) = u(\pi'[\mathcal{S}])$.*

It is possible to consider stronger notions of secure implementation, e.g. requiring that one game allows a given equilibrium using a dominating strategy if and only if the other protocol allows the same equilibrium using a dominating strategy.

4.4.3 Protocols Implementing Equilibria (Game Theory)

Another notion of implementation, very different from the notion of implementing resources, is the notion of a protocol implementing a special type of equilibrium. Given a concrete game $\text{Game} = (\mathcal{R}, \Omega, u, X)$ one can ask whether there exists a protocol $\pi[\mathcal{R}]$ which is an equilibrium and in addition has some desired property, like always letting all parties learn some function $f(x_1, \ldots, x_n)$ of inputs of their choice. This has no *a priori* connection to the notion of a game structure implementing a resource. As an example, an equilibrium allowing all parties to learn $f(x_1, \ldots, x_n)$ might not be secure to use in place of a *function mediator* M_f, which on input $(x_1, \ldots, x_n)$ outputs $f(x_1, \ldots, x_n)$, in some context. The reason is that $\pi[\mathcal{R}]$ implementing a given equilibrium in $\text{Game} = (\mathcal{R}, \Omega, u, X)$ depends on u. If $\pi[\mathcal{R}]$ is used in place of M_f in another context, then this context probably will not assign the same utilities, u, to the evaluation of M_f, as the evaluation of M_f is just part of some larger game. Indeed, the context might not even assign well-defined utilities to the evaluation of M_f, as the utility of the overall game might depend on many other actions than the evaluation of M_f.

In the survey we will distinguish between work on implementing equilibria using protocols and implementing resources using game structures.

4.5 Challenges of Secure Implementation

In this section we discuss our example extensive definition and why it is almost impossible to achieve. We discuss three related issue, *privacy*, *no signaling* and *no correlation*. We will see that any intensive notion implying the extensive notion must guarantee the privacy of inputs, it must disallow coordination between the parties and must even disallow that the parties make correlated actions.

4.5.1 The Need for Privacy

Observe that our extensive definition (Definition 4.5) does not mention any privacy concerns at all. This seems fair in a game theoretic setting where we postulate that games will be played according to equilibria determined by utilities. If privacy is in the utility of the parties, then the concern will arise via the notion of equilibria. If it does not, privacy is not an issue. We will, however, now argue that privacy is a necessary part of an intensive definition.

We illustrate this by sketching a simple intensive definition which requires that γ and $\mathcal{R}$ have the same equilibria. We then show that this is not enough for γ to be an implementation of $\mathcal{R}$ in the extensive sense.

Strategic Form Implementation

To illustrate that privacy is important in an intensive definition, we use the observation that any single game, using any type of resource R, can be implemented in one round using simultaneous broadcast. If the resource R is randomized, then in addition we need a *common reference string* resource CRS, which makes public a uniformly random string.

Consider any game structure $\text{Game} = (\mathcal{R}, \Omega)$. We are going to define another game structure $\text{Game}' = ((SBC, CRS), \Omega')$ allowing the same equilibria; Here SBC is a resource which allows a synchronous broadcast in the first round and then terminates, and CRS is a resource giving, in the first round, a common output $s \in \{0,1\}^*$ where s is a uniformly random string of the length that $\mathcal{R}$ uses for randomness.[8]

The common output Ω' is defined as follows. Let $comm' = ((m_1, \ldots, m_n), s)$ be the common outputs from SBC and CRS after a protocol π' has run in the game structure $\text{Game}' = ((SBC, CRS), \Omega')$. Here m_i is the message broadcast by P_i and s is the common reference string. The message m_i is parsed as a triple (x_i, A_i, r_i). Then x_i is parsed as an input of P_i, A_i is parsed as an IOA, and r_i is parsed as a bit-string of the length that A_i uses for randomness. Then $\Omega'(comm') = \Omega(comm)$, where $comm = \Omega(A_1(x_1; r_1), \ldots, A_n(x_n; r_n), \mathcal{R}(s))$ is the common output computed when running each A_i on input x_i and randomness r_i together with the resource $\mathcal{R}$ run with randomness s.

It is fairly straightforward to see that the game structures Game and Game' allow the same equilibria. To see this, consider any protocol $\pi = (A_1, \ldots, A_n)$ for the game structure $\text{Game} = (\mathcal{R}, \Omega)$. We define a corresponding one-round protocol $\pi' = T(\pi) = (A'_1, \ldots, A'_n)$ for the game structure $\text{Game}' = ((SBC, CRS), \Omega')$. For later use, we add to the description of $T(\pi)$ the computation of the common output. The protocol $\pi' = T(\pi)$ runs as follows:

1. Each P_i has some input x_i.

[8] By (SBC, CRS) we denote the resource which provides the service of both resources SBC and CRS.

2. Each P_i samples a uniformly random bit-string $r_i \in \{0,1\}^*$ of the length that A_i uses for randomness, where A_i is the strategy that P_i has in π.

3. Then each P_i broadcasts (x_i, A_i, r_i), simultaneously with all other parties.

4. All parties receive the common outputs $((x_1, A_1, r_1), \ldots, (x_n, A_n, r_n))$, from SBC, and s, from CRS.

5. Since the parties are playing the game $\text{Game}' = ((SBC, CRS), \Omega', u)$, the common output will be $\Omega'(\pi'(X)) = \Omega(comm)$, where $comm = \Omega(A_1(x_1; r_1), \ldots, A_n(x_n; r_n), \mathcal{R}(s))$.

Since π is for the game structure $\text{Game} = (\mathcal{R}, \Omega)$, we have that $\Omega(\pi(X)) = \Omega(comm)$, where $comm = \Omega(A_1(x_1; r_1), \ldots, A_n(x_n; r_n), \mathcal{R}(s))$ and where each r_i is a uniformly random bit-string $r_i \in \{0,1\}^*$ of the length that A_i uses for randomness and s is a uniformly random bit-string $s \in \{0,1\}^*$ of the length that R uses for randomness. Since the output s from CRS is also a uniform random bit-string of the length that R uses for randomness, the following lemma is immediate.

Lemma 4.1 *Let $\pi = (A_1, \ldots, A_n)$ be any n-party protocol for the game structure* $\text{Game} = (\mathcal{R}, \Omega)$ *and let $\pi' = T(\pi) = (A'_1, \ldots, A'_n)$ be the corresponding protocol for the game structure* $\text{Game} = ((SBC, CRS), \Omega')$. *Then it holds for all input distributions X that $\Omega'(\pi'(X)) \equiv \Omega(\pi(X))$; I.e., the two protocols have the exact same distribution of the common outputs on the same input distributions.*

Note for later use that the transformation $T(\pi)$ works on the code of each party locally. I.e., we can write the transformation as $T(\pi) = (T(A_1), \ldots, T(A_n))$.

Let $\pi' = (A'_1, \ldots, A'_n)$ be any protocol for the game structure $\text{Game}' = ((SBC, CRS), \Omega')$. We define a corresponding protocol $\pi = T^{-1}(\pi') = (A_1, \ldots, A_n)$ for the game structure $\text{Game} = (\mathcal{R}, \Omega)$. For later use, we include in the description of $T(\pi')$ the computation of the outcome:

1. Each P_i has some input x'_i.

2. Each P_i runs $A'_i(x'_i)$ to receive the first message m_i that $A'_i(x'_i)$ would have broadcast in the game structure $\text{Game} = ((SBC, CRS), \Omega')$, and parses m_i on the form (x_i, A_i, r_i).

3. Each P_i participates in the game structure $(\mathcal{R}, \Omega)$ by running $A_i(x_i; r_i)$.

4. Since the parties are playing in the game structure $\text{Game} = (\mathcal{R}, \Omega)$, the common output will be $\Omega(T(\pi')(X')) = \Omega(A_1(x_1; r_1), \ldots, A_n(x_n; r_n), \mathcal{R}(s))$ (where s is the randomness used by $\mathcal{R}$).

We then compare to $\Omega'(\pi'(X'))$, which is computed as follows, by running in the game structure $((SBC, CRS), \Omega')$:

1. Each P_i has some input x'_i.

2. Each P_i runs $A'_i(x'_i)$ to receive the first message m_i to be broadcast, and broadcasts m_i simultaneously with the other parties.

3. All parties receive $(m_1, \ldots, m_n)$ and the output s from CRS.

4. Since the parties are running in the game structure $((SBC, CRS), \Omega')$, to compute the outcome each m_i is then parsed on the form (x_i, A_i, r_i) and the outcome is $\Omega'(\pi'(X')) = \Omega(A_1(x_1; r_1), \ldots, A_n(x_n; r_n), \mathcal{R}(s))$.

This proves the following lemma.

Lemma 4.2 *Let* $\pi' = (A'_1, \ldots, A'_n)$ *be any protocol for the game structure* Game $= ((SBC, CRS), \Omega')$ *and let* $\pi = T^{-1}(\pi') = (A_1, \ldots, A_n)$ *be the corresponding protocol for the game structure* Game $= (\mathcal{R}, \Omega)$. *Then it holds for all input distributions* X *that* $\Omega(\pi(X)) \equiv \Omega'(\pi'(X))$.

Note for later use that the transformation $T^{-1}(\pi')$ works on the code of each party locally. I.e., we can write the transformation as $T^{-1}(\pi') = (T^{-1}(A'_1), \ldots, T^{-1}(A'_n))$. Note also that even though the code of $T^{-1}(T(\pi))$ is not identical to the code of π, it is behaviorly identical. To see this note that $T(A_i)(x_i)$ will broadcast (x_i, A_i, r_i) as the first message. Therefore $T^{-1}(T(A_i))(x_i)$ first runs $T(A_i)(x_i)$ to get (x_i, A_i, r_i) and then runs $A_i(x_i; r_i)$ in the protocol, exactly as $A_i(x_i)$ does. In the same way it is easy to see that $T(T^{-1}(\pi'))$ and π' have the same behavior. Together with the lemmas this allows to prove the following theorem.

Theorem 4.1 *Let* Game $= (\mathcal{R}, \Omega, X, u)$ *be any game where the utility depends only on* $\Omega(\pi(X))$ *(i.e., the inputs and the common outputs), and let* Game$' = ((SBC, CRS), \Omega', X, u)$ *be the corresponding game. Then it holds for all protocols* π *for* Game *that* π *is a Nash equilibrium for* Game *if and only if* $\pi' = T(\pi)$ *is a Nash equilibrium for* Game$'$. *Furthermore,* π *is a dominating strategy for* Game *if and only if* $\pi' = T(\pi)$ *is a dominating strategy for* Game$'$. *And, vice versa, it holds for all protocols* π' *for* Game$'$ *that* π' *is a Nash equilibrium for* Game$'$ *if and only if* $\pi = T^{-1}(\pi')$ *is a Nash equilibrium for* Game, *and* π' *is a dominating strategy for* Game$'$ *if and only if* $\pi = T(\pi')$ *is a dominating strategy for* Game.

Proof: We only give an example of how the theorem is proved. The other cases are handled in similar ways. We prove that if π is a Nash equilibrium for Game then $\pi' = T(\pi)$ is a Nash equilibrium for Game$'$.

So, assume that π is a Nash equilibrium for Game. We have to prove that $\pi' = T(\pi)$ is a Nash equilibrium for Game$'$. For the sake of contradiction, assume that it is not. Therefore there exists a party P_i and a protocol $\hat{\pi}' = (\hat{A}'_i, \pi'_{-i})$ such that

$$u_i(\Omega(\pi'(X))) < u_i(\Omega(\hat{\pi}'(X))) \ . \tag{4.1}$$

Now let $\hat{\gamma} = T^{-1}(\hat{\pi}')$. By Lemma 4.2 we have that

$$u_i(\Omega(\hat{\gamma}(X))) = u_i(\Omega(\hat{\pi}'(X))) \ . \tag{4.2}$$

Since $\pi' = T(\pi)$ we have from Lemma 4.1 that

$$u_i(\Omega(\pi'(X))) = u_i(\Omega(\pi(X))) \ . \tag{4.3}$$

Substituting with (4.2) and (4.3) in (4.1) we get that

$$u_i(\Omega(\pi(X))) < u_i(\Omega(\hat{\gamma}(X))) \ . \tag{4.4}$$

Recall that $\hat{\gamma} = T^{-1}(\hat{\pi}')$, $\hat{\pi}' = (\hat{A}'_i, \pi'_{-i})$ and $\pi' = T(\pi)$. So, if we let $\hat{A}_i = T^{-1}(\hat{A}'_i)$ and use that T and T^{-1} are local transformations, we get that:

$$\begin{aligned}\hat{\gamma} &= T^{-1}(\hat{A}'_i, T(\pi)_{-i}) \\ &= (T^{-1}(\hat{A}'_i), T^{-1}(T(\pi)_{-i})) \\ &= (\hat{A}_i, T^{-1}(T(\pi))_{-i}) \ .\end{aligned}$$

Since π_{-i} and $T^{-1}(T(\pi))_{-i}$ have the same behavior, it follows that if we let $\hat{\pi} = (\hat{A}_i, \pi_{-i})$, then $\hat{\pi}$ and $\hat{\gamma}$ have the same behavior, which in particular means that

$$u_i(\Omega(\hat{\gamma}(X))) = u_i(\Omega(\hat{\pi}(X))) \ .$$

Combining this with (4.4) we get that

$$u_i(\Omega(\pi(X))) < u_i(\Omega(\hat{\pi}(X))) \ ,$$

which is the contradiction to π being a Nash equilibrium for Game, as $\hat{\pi}$ is of the form $(\hat{A}_i, \pi_{-i})$. QED

The above theorem shows that if we adopt an intensive definition of game-theoretic secure implementation in the style of Game$' = ((SBC, CRS), \Omega')$ having the same equilibria as Game for all game parameters, then the above Game$'$ should be considered a game-theoretic secure implementation of Game.

The Need for Privacy

Seen from a cryptographic point of view one would immediately become wary about deeming the above game structure Game$' = ((SBC, CRS), \Omega')$ a secure implementation of Game $= (\mathcal{R}, \Omega)$, at least for most game structures Game. The reason of course being that in Game$' = ((SBC, CRS), \Omega')$, the parties are essentially forced to reveal their inputs, whereas this might not be the case in a game using the resource $\mathcal{R}$.

To illustrate that revealing the types might make a big difference, consider a commercial setting where P_1 has some data, and where P_2 wants to buy some of this data from P_1. In addition, if P_2 can learn more data than agreed upon, it will be an advantage of P_2 and a disadvantage of P_1.

We can idealize this setting as a small weakly mediated game, as follows. Party P_1 has a type (x_0, x_1), where $x_0, x_1 \in \{1, \ldots, N\}$ for $N = 50$, say. Party P_2 has a type $c \in \{0, 1\}$, indicating which data item x_c it needs. For the remaining we assume that x_0 and x_1 are uniformly random and that c is an independent uniformly random bit.

The parties have access to a mediator M where P_1 inputs (x_0, x_1), where P_2 inputs c and where M outputs x_c privately to P_2. After the mediation P_2 then plays an action (y_0, y_1), where we can think of y_0 as its guess at what x_0 is and think of y_1 as its guess at what x_1 is. The utility of P_1 is then defined to be 0 if $x_0 \neq y_0$ and $x_1 \neq y_1$.[9] The utility of P_1 is 100 if $x_c = y_c$ and $x_{1-c} \neq y_{1-c}$, and the utility of P_1 is 50 if $x_0 = y_0$ and $x_1 = y_1$. The utility of P_2 is 100 if $x_c = y_c$ and $x_{1-c} \neq y_{1-c}$. The utility of P_2 is 150 if $x_c = y_c$ and $x_{1-c} = y_{1-c}$. Otherwise, the utility of P_2 is 0.

Consider the protocol where P_1 inputs (x_0, x_1) to M, P_2 inputs c to M, and where P_2 then plays the actions (y_0, y_1) where $y_c = x_c$ is the output it received from M and y_{1-c} is a

[9] In this case P_2 will not pay for the data.

uniformly random value from $\{1, \ldots, 50\}$. It is clear that $x_c = y_c$. If $x_{1-c} \neq y_{1-c}$, then the utility is $(100, 100)$. If by accident $x_{1-c} = y_{1-c}$, then the utility is $(50, 150)$. The utility is therefore $(100 - 50N^{-1}, 100 + 50N^{-1}) = (101, 99)$, and it is easy to see that this is a Nash equilibrium.[10]

Assume now that we implement the mediator M using one round of simultaneous broadcast. This means that the game which is now played is that P_1 and P_2 first broadcast a message m_1 respectively m_2, and then P_2 plays an action (y_0, y_1). The utilities are given from $(x_0, x_1), (y_0, y_1), c$ as before, as the outcome function is such that the first round of broadcast does not influence the outcome of the overall game.

We argue that there is no equilibrium where P_1 has utility larger than 50. Consider any equilibrium obtained using strategy (A_1, A_2). We first argue that the equilibrium could be obtained using another strategy (A_1, A_2'), where A_2' is of a special form. For any message m_1 that P_1 can send in the initial broadcast, let $Y_0(m, c)$ denotes P_2's best guess at x_0 and let $Y_1(m, c)$ denote P_2's best guess at x_1. It is clear that A_2's best strategy is to play $(y_0, y_1) \leftarrow (Y_0(m, c), Y_1(m, c))$, as P_2's payoff is monotone increasing in the number of x_b's it guesses correctly. It is also clear that $Y_0(m, c)$ and $Y_1(m, c)$ are independent of c, as m is independent of c.[11] We can therefore write an optimal strategy of P_2 as $(y_0, y_1) \leftarrow (Y_0(m), Y_1(m))$. Then let p_0 be that probability that $y_0 \neq x_0$ and $y_1 \neq x_1$, let p_2 be that probability that $y_0 = x_0$ and $y_1 = x_1$, and let $p_1 = 1 - p_0 - p_2$ be the probability that $y_b = x_b$ for exactly one $b \in \{0, 1\}$. Finally, let p_c be the probability that $y_c = x_c$ and $y_{1-c} \neq x_{1-c}$. It it easy to see that $p_c = \frac{1}{2}p_1$. Therefore the utility of P_1 is $0p_0 + 100p_c + 50p_2 = 50(p_1 + p_2) \leq 50$, as claimed.

This shows that even though the strategic form game implements the mediator M with exactly the same equilibria, when the implementation is used instead of M in a larger context, the equilibria changes dramatically. In the above case, this was due to the fact that P_1 has to leak unnecessary information about its type, as P_1 must play according to a strategy where with some probability p_2 the party P_2 can compute both x_0 and x_1 and with probability $1 - p_2$ it can compute exactly one of them.

The reason why this problem does not turn up when only M and the strategic form implementation are considered is that in the strategic form implementation the type of each party is hidden until the broadcast. Therefore P_2 cannot pick c based on the inputs in this single game. We say that the strategic form game has *pre-game privacy*. After the game has been played P_2 will however know the type of P_1. We say that the strategic form game does not have *post-game privacy*. In these terms, the pre-game privacy ensures that equilibria are maintain when a single game is played and the lack of post-game privacy prevents the implementation from being secure in a context where more games are played after the execution of the implementation, or where the outcome of a single "outer" game is affected by actions performed after the execution of the supposedly secure implementation of M.

[10] If P_1 inputs (x_0', x_1') where $x_d' \neq x_d$ for some $d \in \{0, 1\}$, then in the case $c = d$, which happens with probability $\frac{1}{2}$, the utility of P_1 will be at most 1, giving a maximal expected outcome of $51 < 99$. On the other hand, if P_2 does not play $y_c = x_c$, then its utility will always be 0. In addition, picking y_{1-c} different from uniformly random will not increase its utility, as x_{1-c} is uniformly random.

[11] The inputs of P_1 and P_2 are independent and P_2 cannot signal to P_1 before P_1 picks m.

4.5.2 No Signaling

The above section showed that privacy is important. Pre-game privacy to not disturb the equilibria of a single game and post-game privacy to not disturb equilibria in the context. When we want an intensive definition to imply the extensive definition, more subtle issues arise too, including the notion of *signaling*. Consider a game where P_1 has a uniformly random type $x_1 \in \{0,1\}$ and P_2 has an independent uniformly random type $x_2 \in \{0,1\}$. In the game P_1 broadcasts a single bit $b_1 \in \{0,1\}$ and P_2 simultaneously broadcasts a single bit $b_2 \in \{0,1\}$. The utility of each party is 1 if $b_1 \oplus b_2 = x_1 x_2$. Since the parties cannot communicate (signal) about there types before b_1 and b_2 are broadcast, it is easy to see that the best equilibrium obtainable is $(3/4, 3/4)$ which is e.g. obtained by playing $(b_1, b_2) = (0,0)$. If however P_1 can signal a single bit to P_2 before the playing of b_1 and b_2, then P_1 can signal x_1 to P_2, and then the best equilibrium is $(1,1)$ (obtained by e.g. playing $(b_1, b_2) = (0, x_1 x_2)$). So, if previous to playing this game a mediator which does not allow P_1 and P_2 to signal has been replaced by an implementation which allows e.g. P_1 to signal a single bit to P_2, then the equilibria of this later game changes.

In general, if a mediator is replaced by an implementation which allows more signaling than the mediator, then later equilibria will change in some contexts. Again we divide in *no pre-game signaling* and *no post-game signaling*. No pre-game signaling is necessary for an implementation to not disturb the equilibria of a single game, and no post-game signaling is needed to not disturb equilibria in the context.

4.5.3 No Correlation

When we want an intensive definition to imply the extensive definition, an even more subtle issue arises, which we will call *no correlation*. It is similar to signaling in that it allows the parties to coordinate their actions but is different in that it does not allow the transfer of information between the parties.

To illustrate the notion of correlation, consider a game where the parties have no types. In the game, P_1 broadcasts a single bit $b_1 \in \{1,2\}$ and P_2 simultaneously broadcasts a single bit $b_2 \in \{1,2\}$. If $b_1 \neq b_2$ then both parties have utility 0. If $b_1 = 1 = b_2$, then P_1 has utility 2 and P_2 has utility 1. If $b_1 = 2 = b_2$, then P_1 has utility 1 and P_2 has utility 2. This is a very idealized model of a setting where P_1 and P_2 get a utility from coordinating their actions, but where the parties have different desired outcomes of the joint work.

Consider a game where first a resource CRS outputs a common uniformly random bit $b \in \{1,2\}$, and then the parties must play their actions simultaneously. It is clear that the strategy where each party plays b is an equilibrium, with utility $(3/2, 3/2)$. Consider then a game where CRS is removed and the parties have to simultaneously broadcast their actions in the first round. It is straight-forward to see that here there is no strategy implementing an equilibrium $(3/2, 3/2)$. In fact, there is no strategy with utility $(3/2, 3/2)$, equilibrium or not. To see this, note that the strategies of the parties must be independent. I.e., P_1 plays $b_1 = 1$ with probability p_1 and P_1 plays $b_2 = 1$ with probability p_2. Therefore the utility is $p_1 p_2 (2,1) + (1-p_1)(1-p_2)(1,2)$. So, if we sum the utilities of the parties, then we get $3(p_1 p_2 + (1-p_1)(1-p_2)) \leq 3$, where equality is obtained only if $p_1 = 1 = p_2$ or $p_1 = 0 = p_2$. Therefore the only utilities with sum 3 are $(2,1)$ and $(1,2)$. Since the sum of utilities in $(3/2, 3/2)$ is 3, this shows that $(3/2, 3/2)$ is impossible.

Notice that the possible equilibria changed even though P_1 and P_2 were allowed no more

signaling. All that they got extra was some correlated randomness. We say that the parties were allowed *correlation*, and the example shows that an implementation must have *no pre-game correlation* to not disturb equilibria in a single game and must have *no post-game correlation* to not disturb equilibria in the context.

We note that correlation can be much more subtle than broadcasting a random bit. Consider e.g. a setting with n parties, where prior to the game P_i is given a uniformly random bit b_i and where the parties know that $\Pr[\oplus_{i=1}^{n} b_i = 1] = \frac{1}{2} + \epsilon$ for some ϵ. If $\epsilon = 0$, then the bits b_i are uniformly random and mutually independent, so they clearly allow no extra correlation, as each P_i could have sampled b_i on his own. If however $\epsilon \neq 0$, then the distribution of the b_i can disturb equilibria in some games played after the distribution of the b_i. We leave it as a small exercise for the reader to contrive an example of this.[12] This shows that even the slightest of extra correlation, in an implementation over the mediator, will render the implementation insecure in some contexts.

4.5.4 Non-Monotonicity

A last general observation about the game-theoretic notion of secure implementation which is convenient to discuss in general terms before looking at the concrete models and techniques is the notion that the power of a network is not monotone in the available resources, which is opposed to the cryptographic notions. In cryptographic protocol theory, introducing more resources into a network never decreases the power of the protocol, in terms of which other resources can be implemented on the network. This is not true in the game-theoretic setting. The trivial reason is that an additional resource might allow more signaling or more correlation.

4.6 Popular Assumptions and Techniques

Before we take a more in-depth view of some of the models and techniques which have been proposed for rational cryptography, we give a brief overview of some of the popular assumptions and techniques which have been used in several works.

4.6.1 Preprocessing

The use of a preprocessing phase was suggested in [LMS04, LMS05] to prevent collusions. This approach assumes that all parties execute a subprotocol during the preprocessing phase in which collusions are possible. This subprotocol still requires that the output of each party remains private, which can be achieved using any protocol for secure function evaluation. It is assumed that until the end of the preprocessing, parties do not know their types yet. After the preprocessing, parties receive their types and run a protocol whose execution is "driven" by the result of the preprocessing. Indeed, the protocol is designed in such a way that parties' actions are forced and no information about the type is detectable by other parties. A preprocessing stage is therefore often combined with forced-action techniques.

[12] Note that the above two-party example is of this general form, with $\epsilon = 0$.

4.6.2 Ballot Boxes

The ballot box is a physical gadget that has been used as a secure mixer in the real world. It is run on input some objects and it outputs a random permutation of those objects. A popular use of the ballot box is that of a tally function. The power of a ballot box has been recently studied in [IML05] where the authors show that any mediated game of incomplete information can be rationally and securely simulated by a ballot-box game. This game is a sequence of actions that use a ballot box and envelopes. For more details, see Section 4.8.

4.6.3 Envelopes and Super Envelopes

Informally, the existence of physical envelopes is an assumption that allows one to implement a commitment scheme where no additional information is transferred between sender and receiver. Notice that a classical commitment computed by means of a randomized cryptographic algorithm allows one to transfer much more information. Indeed, the use of randomness allows one to compute several *correct* messages among which one of them can be chosen to communicate a given information. In general, without setup/physical assumptions, when there is a sufficient amount of entropy in a protocol, then steganographic communications can not be prevented.

More concretely, a physical envelope implements the following procedures.

1. A sender encodes a message m in the envelope that publicly shows the name of the receiver. With this procedure the only information that any party different from the sender can obtain is that there is a message from the sender to the receiver.

2. The receiver opens the envelope and reads m. With this procedure the only information that any party different from the receiver and the sender can obtain is that the receiver has opened the envelope. The only extra information for the receiver is that the sender encoded m in the envelope. The only extra information for the sender is that the receiver has read m.

Such envelopes (possibly with minor variations) have been implicitly used in [Bár92] and explicitly in [LMPS04] and then in [LMS05] where along with a preprocessing phase they obtain collusion-free protocols. Moreover, they have been used in [IML05] where it is shown that super-envelopes (envelopes that contain other envelopes) and a ballot-box allow to rationally and securely simulate any mediated game of incomplete information.

Several variants of this envelope functionality have been proposed. For instance in [MN05, MN06] envelopes with tamper-evident seals, which constitute a relaxation to the functionality described above, have been proposed with the following properties:

- Nobody is notified when a player holding an envelope opens it (i.e., breaks the seal);
- When a player holds an envelope, he can check whether or not its seal was broken.

A common physical example of an envelope with tamper-evident seals is that of scratch-off cards, which are widely used in lotteries for instance. There are several natural variants of the functionality for envelopes with tamper-evident seals:

- Distinguishability vs. indistinguishability of sealed envelopes: can the sender distinguish his own envelopes from envelopes sent by other parties when seals are not broken?

- Breakability vs. unbreakability of seals: do we assume that honest users can break seals?

Moran and Naor [MN05] investigate the possibility of realizing classical cryptographic primitives by using tamper-evident sealed envelopes, that is, in the "tamper-evident sealed envelopes model". As a result, they show that it is impossible to realize oblivious transfer when breakable seals are considered, while coin tossing and bit commitment can be realized. Also, when distinguishable envelopes with unbreakable seals are considered, bit commitment becomes impossible.

4.6.4 Point-to-Point Channels vs. Broadcast Channels

Whenever parties are concerned by a cryptographic protocol, there is a fundamental distinction regarding their communication medium. The basic case is that parties are connected by pairwise point-to-point channels, which may additionally be authenticated or private, i.e., support integrity protection or confidentiality protection. When communication uses only pairwise channels among a group of potentially faulty or malicious parties, however, not all parties necessarily observe the same messages. In this situation, merely reaching agreement on a single bit among the non-faulty parties represents already the difficult problem of *Byzantine agreement* [PSL80]. The difficulty is caused by potentially faulty parties who might send conflicting messages over the pairwise channels.

A *broadcast channel* in the synchronous model (see Section 4.6.5) is an abstraction which guarantees that all non-faulty parties receive the same set of messages through the channel and that they receive them in the same order. It can be implemented using a protocol for Byzantine agreement.

4.6.5 Synchrony

Another popular assumption is that of *synchronized clocks*. In a *synchronous network*, there are known upper bounds on the delay of messages on any channel and on the relative clock speeds of the parties. For simplicity and without loss of generality, one can imagine that all communication in a synchronous network occurs in a sequence of global rounds using either the point-to-point channels or the broadcast channel. In every round, every party sends a message on all point-to-point channels and at the end of the round, every non-faulty party receives a message on all point-to-point channels or a special symbol $\perp$ indicating that no message was sent. Similarly for the broadcast channel, in every round one distinguished party sends a message on the broadcast channel (or is at least supposed to send one), and all parties receive that message or a special symbol $\perp$ indicating that no message was sent. The sequence of senders is predefined.

A primitive that combines both synchronized clocks and synchronous broadcast is the *simultaneous broadcast channel*. This primitive assumes that in a given time slot all parties broadcast their messages. Then each party receives all messages sent by the other parties. The guarantee of this primitive is that no party is able to first receive one of the messages sent by the other parties and then send his message without being detected. Many cryptographic protocols for multi-party computation assume this idealized model.

In an *asynchronous network*, no common clock exists and the delay of all messages on the point-to-point channels is unbounded. Typically an adversarial scheduler is assumed to control the delivery of messages over the network. The counterpart of a synchronous broadcast channel is an asynchronous atomic broadcast facility. Implementing asynchronous atomic

broadcast and asynchronous Byzantine agreement requires randomized protocols and all such protocols are subject to a negligible probability of failure. This is because of the "FLP" impossibility result which shows that all deterministic agreement protocols in asynchronous networks with even a single faulty party have non-terminating executions [FLP85].

In [LT06] (see Section 4.10), it is shown how to use incentives for parties so that they rationally use true inputs. This allows one to obtain protocols where rational parties do not deviate from the prescribed behavior. Their construction is based on a simultaneous broadcast channel and tolerates additional channels between parties, in contrast to [IML05] where moreover envelopes and a ballot box are also needed. The results obtained in [HT04] (see Section 4.9) need both a simultaneous broadcast channel and private channels. A simultaneous broadcast channel is also used in [GK06] (see Section 4.9.3).

4.6.6 Forced Actions

Another popular technique is that of *forced action*. This approach consists in forcing the actions of the parties in a verifiable way, so that they can only run the prescribed protocol and thus can not transfer any steganographic information. This roughly means that players have to play in a predetermined order and in specific time steps to prevent the transfer of steganographic communication. Forced actions are crucial for the protocol of [IML05] that achieves rational secure computation.

Very popular tools for designing secure protocols are proof systems with special security requirements as witness indistinguishability [FS90] and zero-knowledge [GMR89]. Assume that there already are setup/physical assumptions that remove the steganographic communication encoded by prover and verifier in their messages. The possibility that a prover uses a given witness w instead of another witness w' for computing the proof can potentially be used to transfer one bit of information from the prover to the verifier. This problem has been studied in [LMS04] where the concept of unique zero-knowledge (*uniZK*, in short) has been proposed and a construction based on number-theoretic assumptions for multi-theorem non-interactive unique zero-knowledge is given on top of a preprocessing stage. In [IML05] a construction for bounded non-interactive unique zero-knowledge is given on top of the existence of one-way permutations.

4.7 Implementing Correlated Equilibria

Lepinski *et al.* [LMPS04] consider implementing correlated equilibria for strategic-form games using so-called cheap-talk. In a non-cooperative game in strategic form each P_i independently chooses an action $a_i \in A_i$ and privately outputs this action. The utility of P_i is then computed as $u_i(a_1, \ldots, a_n)$.[13] A correlated equilibrium for u is a distribution $E = (r_1, \ldots, r_n) \in A_1 \times \cdots \times A_n$, where r_i is the recommendation of P_i. The corresponding game is played using a resource $\mathcal{R}_E$ which samples $(r_1, \ldots, r_n) \leftarrow E$ and outputs r_i privately to P_i.

In [Bár92] Bárány proposed replacing $\mathcal{R}_E$ by so-called *cheap talk*, where the parties instead of using $\mathcal{R}_E$ will engage in some protocol. This protocol might ideally involve only the use of point-to-point lines, but might use any resource $\mathcal{R}$. For a resource $\mathcal{R}$ the game proceeds as follows:

[13] We can let u_i implicitly interpret any value $a_i \in \{0,1\}^*$ as a valid action $a_i \in A_i$. We can therefore in the following consider u as the full specification of the strategic-form game.

1. The parties interact through $\mathcal{R}$.
2. Based on its interaction with $\mathcal{R}$ each P_i picks an action $a_i \in A_i$ and outputs a_i.
3. The utility of P_i is then computed as $u_i(a_1, \ldots, a_n)$.

We call this the cheap-talk game $(\mathcal{R}, u)$. Note in particular that the utility is independent of the interaction with $\mathcal{R}$ — the utility only depends on the actions taken; This is the reason for the term cheap-talk: it has no implications, except that it allows the parties to correlate their actions.

A special case of a cheap-talk game is the *ideal cheap-talk game* $(\mathcal{R}_E, u)$ which uses the resource for sampling $E = (r_1, \ldots, r_n) \in A_1 \times \cdots \times A_n$. For this ideal cheap-talk game we use δ to denote the *dummy protocol* where each P_i outputs $a_i = r_i$. By definition E is a correlated equilibrium for u if the dummy protocol δ is a Nash equilibrium for the ideal cheap-talk game $(\mathcal{R}_E, u)$.

Bárány proposed a protocol for implementing any $\mathcal{R}_E$ using enhanced point-to-point lines, which can be modeled using some resource $\mathcal{R}_{epp}$.[14] More precisely Bárány showed that for any correlated equilibrium E there exists a protocol π for $(\mathcal{R}_{epp}, u)$ which is a Nash equilibrium and has the same utility as (E, u). Unfortunately the protocol is vulnerable to coalitions, allowing just two colluding parties to force π to give outputs giving them benefit on behalf of the other parties.

4.7.1 Coalition-Safe Cheap Talk

In [LMPS04] Lepinski *et al.* define the notion of *coalition-safe cheap talk* to mean a cheap-talk game which is not vulnerable to the problem discussed in relation to Bárány's protocol above. In particular, it should hold that any size of coalition should not be able to obtain more in the cheap-talk game $(\mathcal{R}, u)$ than in $(\mathcal{R}_E, u)$. More precisely:

Definition 4.6 *Let E be a correlated equilibrium for u, let $(\mathcal{R}_E, u)$ be the corresponding ideal cheap-talk game, and let δ be the dummy protocol for $(\mathcal{R}_E, u)$. Let $\mathcal{R}$ be some resource and let π be a protocol for the cheap-talk $(\mathcal{R}, u)$. We say that π is a* coalition-safe implementation *of E if for any coalition $A \subset \{1, \ldots, n\}$ and any joint strategy π_A for A in $(\mathcal{R}, u)$ there exists a joint strategy δ_A for A in $(\mathcal{R}_E, u)$ such that δ' has the same utility in $(\mathcal{R}_E, u)$ as π' has in $(\mathcal{R}, u)$, where $\delta' = (\delta_{-A}, \delta_A)$ and where $\pi' = (\pi_{-A}, \pi_A)$.*

The joint strategies γ_A and π_A are allowed to use arbitrary side channels to coordinate their actions. In [LMPS04] the authors prove the following theorem.

Theorem 4.2 *It holds for all u and all correlated equilibria E for u that there exists a protocol π which is a coalition-safe implementation of E using the resource $\mathcal{R}$ for ideal envelopes.*

In [LMPS04] more detailed definitions are given, to allow also protocols using cryptographic tools like encryption. For clarity we have left out these details. To get a feeling for the notion of implementation defined in Definition 4.6 we sketch the protocol used by [LMPS04] to prove Theorem 4.2.

[14] *Enhanced* e.g. involving that it is possible to show to other parties what was received from a particular party.

Ideal envelopes model physical envelopes with such properties as: Each envelope has a visible distinct identifier; envelopes hide their contents perfectly when sealed; and envelopes reveal their entire contents when opened. For more on ideal envelopes, see Section 4.6.3.

The protocol used to prove Theorem 4.2 is based on an SFE protocol from [GMW87] tolerating up to $n-1$ corruptions (see Chapter 3 for more details on the protocol from [GMW87]). This protocol has the property that if no party deviates from the protocol, then all parties learn the correct output (and nothing else) and that if some party deviates from the protocol, then all parties will agree on the identity of a deviating party (using the terminology of [LMPS04] the protocol is *abort-identifying*). The protocol however has the problem that not all parties might learn the output. In fact, all protocols not based on physical assumptions must have this property [Cle86]. In [LMPS04] this problem is handled by an intricate protocol using the envelopes to augment the run of the protocol from [GMW87]. After this augmentation the protocol has the property that either all parties learn the output or no party learns anything about the output, called a *completely fair* protocol.

The cheap-talk protocol from [LMPS04] then proceeds as follows. First it runs a completely fair SFE protocol implementing a sampling of $(r_1, \ldots, r_n) \leftarrow E$ where P_i, and only P_i, learns r_i. If this protocol does not abort, then P_i plays $a_i = r_i$. If the protocol fails, then a party which deviated from the protocol is identified, P_n say. Then the remaining parties $P_1, \ldots, P_{n-1}$ run a completely fair SFE protocol implementing a sampling $(r_1, \ldots, r_n) \leftarrow E$ where P_i, and only P_i, learns r_i. The difference from the first run is that P_n does not receive its recommendation r_n, as P_n is no longer taking part in the computation. The protocol continues like this until some subset of the parties have a non-aborting run. Then the parties receiving their recommendations play these. The remaining parties play any action. The reason why this is secure is essentially that the colluding parties cannot use the prefix of aborting runs to bias the recommendation, as the protocol used to implement $\mathcal{R}_E$ has the property that the decision on whether to abort or not must be taken independently of the output that the protocol would have delivered if no abort had occurred. Therefore the colluding parties cannot bias the output of the protocol run that succeeds.

4.7.2 Conclusion

The notion of implementation proposed by [LMPS04] is rather weak compared to some of the notions covered in later chapters. First of all, [LMPS04] only allows to talk about implementing the resource $\mathcal{R}_E$ sampling a correlated equilibrium for a strategic-form game, as opposed to implementing a resource $\mathcal{R}$ which can be used securely in an arbitrary context. Secondly, the notion only requires that the implementation is a Nash equilibrium. This is opposed to other notions where it is required that also the type of equilibrium is maintained, like a dominating strategy in the ideal game becoming a dominating strategy in the implementation too. Also, the protocol might allow the coalition to signal to the non-colluding parties e.g. through which colluding parties abort and when; This might create new Nash equilibria, which is however allowed by Definition 4.6 as it is only required that the specific proposed protocol π can be simulated in the ideal cheap-talk game. Finally, the notion of implementation does not consider games where the parties have secret types.

Despite these limitations, the notion is very interesting as correlated equilibria for strategic form games are a well-studied and well-understood notion. Implementing these using realistic resources is therefore an interesting problem. For this reason the problem of implementing correlated equilibria has been studied by many researchers, including [Ben98, DHR00, Tea04].

We focused on [LMPS04] as it currently seems to provide the highest level of security and efficiency: The protocols in [Ben98, Tea04] do not protect against coalitions. The paper [DHR00] considers essentially the same notion of implementation and also implements $\mathcal{R}_E$ using secure function evaluation. The paper [DHR00] however only considers two-player games. This allows them to use so-called min-max punishment to deal with abortion and allows them to ignore the issue of coalitions. The paper [DHR00] does not use any physical primitives like envelopes.

4.8 Rational Secure Function Evaluation

In [IML05] Izmalkov, Micali and Lepinski propose a notion of *rational secure computation*. In our terminology they consider the problem of securely implementing a function mediator M_f for a strongly mediated game. They want the implementation to be secure to use in any protocol which makes multiple uses of M_f, once in each round say. Their extensive definition therefore seems to be that the implementation of M_f is secure to use multiple times in any context where it is secure to use M_f multiple times. They only consider randomized function mediators. I.e., in each round M_f takes an input $(x_1, \ldots, x_n)$ and outputs a common output $y = f(x_1, \ldots, x_n; r)$ for a uniformly random string r. In particular, the mediator M_f keeps no state.[15] Finally, the authors assume that the context where M_f is used has dominating penalized abort.

In [IML05] both an intensive definition is given, which is claimed to imply the above extensive definition, and an implementation is given for all mediators. A central notion in the intensive definition is that an implementation must be *forced-action*, in the sense that in each round, each party has only one allowed action. All deviations are considered aborts. Since aborts cannot be part of an equilibrium, forced-action ensures that all equilibria have no pre-game signaling and no post-game signaling; If at any step a party had two different legal actions, then the choice of action could be used to signal a bit without causing an abort.

In [IML05] it is shown that any mediator has a forced-action implementation by n parties sitting around a table using envelopes and ballot boxes. Quoting [IML05]: *A ballot-box game is an extensive-form game of incomplete information. It includes physical actions such as sealing a message into an envelope, packing up to* 5 *envelopes into a single, larger one, and randomizing the order of a set of envelopes via a bingo-like device,* the ballot box*. It ends with a special* swap *operation, in which all players simultaneously exchange specified sets of sealed envelopes. Each player then learns the outcome by opening all the envelopes in his possession.* For more on envelopes and ballot boxes, see Section 4.6.

4.8.1 Rational Secure Function Evaluation

In our terms, the proposed intensive definition is as follows.[16] Consider any randomized function $y = f(x_1, \ldots, x_n; r)$, and let $\text{Game}_{\texttt{ideal}}$ be the strongly mediated game where the parties have access to the function mediator M_f. Let $\text{Game}_{\texttt{impl}} = (\mathcal{R}, \Omega)$ be any game structure. Game $\text{Game}_{\texttt{impl}}$ is said to game-theoretic securely implement f if the following holds:

[15] They do consider a setting with private outputs, which we leave out here for simplicity.

[16] We flesh out the definition a little more than in [IML05] to be able to have a fully defined notion of forced-action for any protocol using any resource.

common abort detection: The function Ω and the resource $\mathcal{R}$ is such that any protocol run $\pi[\mathcal{R}](X)$ is divided into *aborted runs* and *completed runs*, where an aborted run is indicated by common output $\Omega = (\texttt{abort}, P_i)$ for some P_i and a completed run is indicated by common output $\Omega = (\texttt{complete}, y)$ for some message y.

input-output structure: The function Ω is such that any protocol run for $\text{Game}_{\texttt{impl}}$ is divided into the first I rounds, which together are called the *input phase*, and a final O rounds, which together are called the *output phase*. The numbers I and O are constant, and if any P_i inputs to $\mathcal{R}$ in any round after round $I + O$, then $\Omega = (\texttt{abort}, P_i)$ for the first P_i to do so.

report commitment: The function Ω and the resource $\mathcal{R}$ are such that if any protocol $(A_1, \ldots, A_n)$ completes the first I rounds without an abort (as defined by **common abort detection**), then by inspecting only the communication that A_i had with $\mathcal{R}$ it is possible to define a unique *possible report* X_i of P_i. Furthermore, by inspecting $\mathcal{R}$ it is possible to define a unique *possible randomizer* r, and for a random run of the protocol, r is uniformly random and independent of the possible reports X_i.

compliance feasibility: The function Ω and the resource $\mathcal{R}$ is such that it holds for all parties P_i and all X_i that P_i has at least one strategy, which is independent of the strategies of the other parties, that will lead to P_i having possible report X_i after I rounds if there are no aborts, and which will not lead to a common output $\Omega = (\texttt{abort}, P_i)$.

output correctness: The function Ω and the resource $\mathcal{R}$ are such that if any protocol $(A_1, \ldots, A_n)$ completes $I + O$ rounds without an abort, then the common output y computed by Ω is $y = f(X_1, \ldots, X_n; r)$ for the values defined by **report commitment**.

forced action I: The function Ω and the resource $\mathcal{R}$ are such that $\mathcal{R}$ has no private outputs in the first I rounds and such that any protocol $(A_1, \ldots, A_n)$ which completes the I first rounds without an abort gives rise to *exactly* the same common outputs from $\mathcal{R}$.

forced action II: The function Ω and the resource $\mathcal{R}$ are such that any protocol $(A_1, \ldots, A_n)$ which completes the $I + O$ first rounds without an abort gives rise to outputs from $\mathcal{R}$ in the last O rounds which are distributed according to a distribution D_y depending only on $y = f(X_1, \ldots, X_n; r)$.

The requirement of **compliance feasibility** is trivial, as a party should have the possibility of picking any report and not aborting, as it has in $\text{Game}_{\texttt{ideal}}$.

Consider then **forced action I**, where it is required that $\mathcal{R}$ gives only constant common outputs during the input phase, if there is no abort. This essentially means that as long as there is no abort, each P_i must be forced to perform actions that make $\mathcal{R}$ give some fixed outputs. In a ballot-box game, for instance, a party could only be allowed to put a specific number of envelopes on the table at a specific time. Any deviation must cause an abort of the protocol. Since a party can always comply with the game to not cause the abort, **forced action I** essentially means that a party has *one* and *only one* allowed visible action at each point in the game during the input phase. The requirement **forced action I** ensures that the visible actions of the parties cannot depend on their inputs, which ensures no pre-game signaling and pre-game privacy. Since the outputs from $\mathcal{R}$ must be *constants*, **forced action**

I also ensures no pre-game correlation. All in all, **forced action I** ensures that the parties are running mutually independent strategies during the input-phase, when there is no abort.

The requirement **report commitment** is then made to ensure that the unique possible reports X_i are chosen by the parties themselves at a point in the protocol where they have been running mutually independent strategies. This ensures that the only strategy that a party P_i has is to either force an abort, or pick a report X_i based only on its type x_i and then pick and run a strategy (given by **compliance feasibility**) which leads to P_i having possible report X_i after round I.

The requirement of **output correctness** is then made to ensure that the outcome of the game is the correct value $y = f(X_1, \ldots, X_n; r)$ computed from the independently chosen reports X_i and the independent uniformly random r, exactly as in the strongly mediated game $\text{Game}_{\texttt{ideal}}$.

The first six requirements together therefore ensure that the implementation allows the same equilibria as the mediator.[17]

The requirement **forced action II** is then made to try to ensure that in any context having access to the function mediator M_f *multiple* times, using $\text{Game}_{\texttt{impl}}$ instead of M_f will not give rise to different equilibria. In particular, the requirement is made to ensure that the execution of the output phase does not leak any information but the value $y = f(X_1, \ldots, X_n; r)$. This follows, as all parties learn, and should learn, y and therefore could have sampled D_y themselves. Therefore **forced action II** guarantees that the output phase does not allow more signaling than does y and guarantees that no more information about the x_i is leaked than when using M_f. I.e., **forced action II** ensures post-game privacy and no post-game signaling.

Note, however, that **forced action II** does *not* ensure no post-game correlation. Since all parties see the common outputs from $\mathcal{R}$ in the output phase, if D_y is not a constant, then D_y is a random value known by all parties, which might allow extra correlation in a later game, as discussed in Section 4.5.3. Therefore the intensive definition does not imply the extensive definition suggested above (and which seems to be the extensive definition suggested by the authors).

As an example that **forced action II** does not ensure no post-game correlation. , one can consider the protocol from [IML05], which is an example of a protocol meeting the above intensive definition. At some point in this protocol, during the output phase, five envelopes (with the numbers $1, \ldots, 5$ in them, respectively) are put in a ballot-box which is then shaken and emptied, and then the five envelopes are opened, to reveal a uniformly random permutation on five objects. This takes place after the input phase, to avoid pre-game correlation, but will allow post-game correlation as all parties now agree on the random permutation.

It seems hard to modify the protocol from [IML05] to avoid post-game correlation. However, it seems easy to deal with this issue by ways of definition. We can e.g. call a game using M_f (several times) *correlation insensitive* if it holds for all $l : \mathbf{N} \to \mathbf{N}$ that the equilibria does not change if in each round r of the protocol we add a resource CRS which outputs a common uniformly random string of length $l(r)$. This would namely after each execution of M_f, with output y, allow the parties to sample D_y using the common random string, showing that having access to a common sampling of D_y does not change the equilibria. We could

[17] There is one subtlety to be considered, as the parties in $\text{Game}_{\texttt{impl}}$ might be given the possibility to abort during the input phase, whereby y will never become known. It should therefore be the case that the utility in case of an abort can be computed based only on which parties aborted. Note, however, that this extra freedom in when to abort cannot lead to new equilibria, as the game is assumed to have dominating penalized abort.

then make the extensive definition that a game structure implements M_f if replacing M_f by the game structure in any correlation insensitive context does not disturb the equilibria. It seems that the above intensive definition implies this slightly weaker extensive definition.

Another, more challenging, approach would be to change the intensive definition by replacing **forced action II** by:

forced action II': The function Ω and the resource $\mathcal{R}$ is such that any protocol $(A_1, \ldots, A_n)$ which completes the $I + O$ first rounds without an abort gives rise to outputs from $\mathcal{R}$ in the last O rounds where the outputs seen by P_i has a distribution $D_{y,i}$ given only by y and i and where the distributions $D_{y,1}, \ldots, D_{y,n}$ are mutually independent given y.

This would ensure that the parties could have simulated the values seen in the output phase by P_i independently sampling $D_{y,i}$ after y becomes known. This intensive definition seems to imply the stronger extensive definition, at least as long as the executions of M_f in the context are performed in sequence.

4.8.2 The Protocol

Going into the details of the protocol from [IML05] would be beyond this survey, but it is worth giving a few general comments on the implementation.

The protocol from [IML05] uses several non-digital primitives and carefully describes the real life setting in which the game must be played. Specifically the authors make use of envelopes, super-envelopes and ballot-boxes (see Section 4.6). The setting is that of all players sitting around a large round table. Further, Barrington's[Bar86] bit and operator representation method with permutations in S_5 is employed to allow for an elegant method for encoding information with sets of 5 envelops.

The implementation begins with a detailed description of how a value is shared amongst all players and how shares can be recombined to reproduce the secret value. This method of sharing values is called a *permutation labeling*. Next 4 (forced action) sub-protocols are described for the following tasks:

- Publicly proving equality of two encoded bits in zero-knowledge,
- Publicly cloning an encoded permutation in zero-knowledge,
- Publicly proving that an encoded permutation encodes either 1 or 0 in zero-knowledge
- Publicly multiplying two encoded permutations in zero-knowledge.

These sub-protocols proceed by parties putting envelopes on the table, opening envelopes on the table, putting envelopes in super-envelopes and ballot-boxes, and shaking and opening ballot-boxes.

Because these sub-protocols are forced action and because all the primitives involved are perfect in an information theoretical sense these operations can not be used to hide subliminal channels. Finally the authors describe in detail the exact order of operations leaving no room for choices when performing the following protocol:

1. Represent the function f as a logic circuit.
2. Given x_i as input, player P_i calculates a permutation labeling for each input wire in the circuit for f and distributes all shares (envelopes) amongst the other players.

3. For each gate g in the circuit for f the players use the previous 4 sub-protocols and their shares of the permutation labeling of g's input wires to evaluate a new permutation labeling encoding the bit on g's output wire in such a way that the correctness of the calculation is publicly verifiable, but it's input and output values remain secret.

4. When all gates in the circuit for f have been evaluated, each player is given the envelopes encoding the value of its output wires. This is done using a *simultaneous swap* operation to ensure fairness.

Since all moves are again forced action and their order is exactly defined, there is no room for stenography and it can be concluded that this is a collusion-resistant implementation of SFE.

4.8.3 Conclusion

The notion of implementation in [IML05] is that of a game structure implementing a resource. The authors make the assumption on the game structure and utility functions that they have dominating penalized abort. They implement any randomized function mediator M_f for a context where M_f is used multiple times. A relatively simple intensive definition is given, which implies a slight weakening of the strongest possible extensive definition, implying security in any correlation insensitive context.

4.9 Non-Cooperative Secret Sharing and Function Evaluation (Purely Rational)

In [HT04] Halpern and Teague consider implementing an n-party function mediator M_f, $f : (\{0,1\}^*)^n \to \{0,1\}^*$ using secure point-to-point channels and simultaneous broadcast. They only consider games where the utilities of the parties are such that they prefer to get the result of the computation over not getting it and where they prefer that as few as possible of the other parties get the output.

4.9.1 Non-Cooperative Secret Sharing

The authors study the special case of secret sharing in some detail. An n-party *secret sharing* scheme[Sha79] with *threshold* t gives for each secret x a distribution $[x]$ over tuples $(x_1, \ldots, x_n)$, such that seeing any t of the shares x_i gives no information on the secret x and given $t+1$ shares x_i one can always compute x; Pooling $t+1$ shares and computing the secret from them is called *reconstruction*. Secret sharing is a notion developed for the Byzantine framework, where by picking t larger than the number of corrupted parties and $t+1$ no larger than the number of honest parties, privacy and reconstruction is always ensured.

The authors motivate their study by the observation that no fixed-round game can implement reconstruction using a dominating strategy when it is a secondary advantage of all parties to withhold the output from the other parties.

Let us first formalize the problem of reconstruction as a game in our framework, following the lines of [HT04]. The main deviation from [HT04] we be due to the paper's using a notion of some party P_i knowing x. It is not formalized what this means, but since the authors consider a computational setting (they e.g. use signatures), it seems reasonable that their notion of knowing x somehow should involve P_i being able to *compute* x *efficiently*. We have

therefore decided to formalize this concept of knowing by requiring P_i to broadcast some guess y_i on x in the last round, and we then say that P_i knows x if $y_i = x$. The *reconstruction game* therefore proceeds as follows. The types $(x_1, \ldots, x_n)$ are sampled according to a secret-sharing scheme $[x]$. Then the parties communicate for a number of rounds, and in some agreed upon last round, each P_i gives a private output y_i. For each P_i a bit s_i is then defined, where $s_i = 1$ if and only if $y_i = x$, and the outcome of the game is $\vec{s} = (s_1, \ldots, s_n)$. Consider two outcomes $\vec{s} = (s_1, \ldots, s_n)$ and $\vec{s}' = (s'_1, \ldots, s'_n)$. That P_i gets less information in the second outcome can be expressed as $s'_i < s'_i$. That P_i gets at least the same information can be expressed as $s'_i \geq s_i$. We can then express the assumptions on the utilities as follows.

U1: For all P_i: The utility is given by $u_i(\vec{s})$. *I.e., the utility is given by which parties get the output.*

U2: For all P_i: If $s_i > s'_i$, then $u_i(\vec{s}) > u_i(\vec{s}')$. *I.e., learning the output is always an advantage for a party, independent of who else receives the output.*

U3: For all P_i: If $\vec{s} \leq \vec{s}'$ and $s_i = s'_i$ and $s_j < s'_j$ for at least one party $P_j \neq P_i$, then $u_i(\vec{s}) > u_i(\vec{s}')$. *I.e., withholding the output from some other party (without getting less information itself) is always an advantage for a party.*

We then say that a protocol implements *non-cooperative reconstruction* if the protocol is an IEDSS[18] in which all parties learn the secret (i.e., $\vec{s} = (1, 1, \ldots, 1)$).

As an example, consider the game where shares are signed, to allow to distinguish good shares from bad shares, and where the parties simultaneously broadcast their shares in the first round and then compute and output x in the second round. If $t < n - 1$, then this is a Nash equilibrium as no party can withhold x from any other party by not broadcasting the good share. It is however not an IEDSS. To see this, consider from the view point of P_1, the joint strategy where P_1 broadcasts x_1 and where only some d of the other parties do so. Consider the alternative strategy for P_1 where P_1 does not broadcast x_1. In both case, if $d \geq t$, then P_1 knows $d + 1 \geq t + 1$ shares and can reconstruct, and so can all other parties. If $d < t$, then P_1 knows at most $d + 1 \leq t$ shares and cannot reconstruct, and neither can the other parties. If $d = t$, then P_1 knows $t + 1$ shares and can reconstruct, but the t parties to broadcast their shares know only the t shares which were broadcast, and thus cannot reconstruct. Since P_1 has higher utility when some other parties do not get x, it follows that P_1 for some strategies of the other parties has an advantage in not broadcasting its own share, and that P_1 never has a disadvantage in not broadcasting. So, the strategy where P_1 broadcasts x_1 is weakly dominated by the strategy where P_1 does not broadcast x_1, showing that broadcasting x_1 is not an IEDSS. So, the above protocol is not a non-cooperative reconstruction. On the other hand, and for the same reasons, the strategy where no party broadcasts anything *is* an IEDSS (and even dominating strategy). Here, however, no party learns the output, so again this protocol is not a non-cooperative reconstruction.

Halpern and Teague [HT04] show that in fact no fixed-round protocol[19] can implement non-cooperative reconstruction, and on the positive note, it is showed that there exists a (non-fixed-round) protocol implementing non-cooperative reconstruction.

[18] More details on this concept is given in Section 4.2.8 and Section 4.10

[19] A protocol where the number of rounds is known before the protocol is run

4.9.2 Non-Cooperative Function Evaluation

The model for secret sharing can be generalized to consider function evaluation. The types $(x_1, \ldots, x_n)$ are sampled from some distribution X, and some function $y = f(x_1, \ldots, x_n)$ is given. Then *pieces of interesting information* are introduced. More formally, a piece of interesting information is a function $I : (\{0,1\}^*)^n \to \{0,1\}^*$, and for any input $(x_1, \ldots, x_n)$, $I(x_1, \ldots, x_n)$ is the specific value of the interesting information on these inputs. Many pieces of information $I_0, I_1, \ldots, I_L$ might be interesting, but it is assumed that the output $I_0 = f$ is always interesting. At the end of the protocol each P_i outputs a vector $\vec{y_i} = (y_{i,0}, \ldots, y_{i,L})$ and a bit-vector $\vec{s_i} = (s_{i,0}, \ldots, s_{i,L})$ is defined, where $s_{i,l} = 1$ if and only if $y_{i,l} = I_l(x_1, \ldots, x_n)$. The outcome of the game is then $\vec{s} = (\vec{s_1}, \ldots, \vec{s_n})$. The authors then make the following assumptions on the utilities of the game.

V1: For all P_i: The utility is given by $u_i(\vec{s})$. *I.e., the utility is given by which parties get which information.*

V2: For all P_i: If $\vec{s_i} \geq \vec{s_i'}$ and $\vec{s_j} \leq \vec{s_j'}$ for all P_j, then $u_i(\vec{s}) \geq u_i(\vec{s'})$. *I.e., if the other parties do not get more information, then P_i gets no less utility by getting more information.*

V3: For all P_i, P_j, $P_i \neq P_j$: If $\vec{s_j} \leq \vec{s_j'}$ and $\vec{s_k} = \vec{s_k'}$ for all $P_k \neq P_j$ and $u_j(\vec{s}) < u_j(\vec{s'})$, then $u_i(\vec{s}) > u_i(\vec{s'})$. *I.e., if P_j gets less utility by being withholding some information, then withholding this information gives P_i more utility.*

V4: For all P_i: If $\vec{s_i} = \vec{s_i'}$ and $\vec{t_i} = \vec{t_i'}$, and it holds for all $P_j \neq P_i$ that $\vec{s_j} = \vec{t_j}$ and $\vec{s_j'} = \vec{t_j'}$, and $u_i(\vec{s}) < u_i(\vec{t})$, then $u_i(\vec{s'}) < u_i(\vec{t'})$. *I.e., whether P_i is better off with certain information is independent of what the other parties have.*

We then say that a protocol implements *non-cooperative evaluation of f* if the protocol is an IEDSS in which all parties learn the secret (i.e., $s_{i,0} = 1$ for all P_i).

The authors prove that if f is non-constant, then no fixed-round protocol implements non-cooperative evaluation of f, and then show that if f is in NCC, then there exists a (non-fixed-round) protocol implementing non-cooperative evaluation of f using simultaneous broadcast.

The argument for the negative result is equivalent to the one given for secret-sharing: Withholding the last message that P_i should have sent will not make it harder for P_i to better evaluate f, but might make it harder for the other parties to evaluate f. Therefore P_i does not send its last message in any IEDSS.

The positive result is essentially optimal, as the class NCC is the class of Boolean functions which can be computed using *some* resource $\mathcal{R}$ (which may be picked specifically for implementing f) when the utilities are of the non-cooperative form above. Shoham and Tennenholtz [ST05] called this class the *non-cooperatively computable* (*NCC*) functions and gave a characterization. Since no function outside NCC has a non-cooperative evaluation using *any* resource, of course they have no implementation using simultaneous broadcast.

To see that function easily can fail to be in NCC, consider the XOR function $y = x_1 \oplus x_2$, and consider the resource with common output $Y = X_1 \oplus X_2$ on inputs X_1 from P_1 and X_2 from P_2. The party P_1 must input some X_1, and in all cases learns $X_2 = Y \oplus X_1$. So, P_1 has no incentive to pick X_1 dependent on x_1. It gets X_2 in all cases, and by picking X_1 independent of x_1 it makes it maximally hard for P_2 to compute $x_1 \oplus x_2$. It follows that any

equilibrium has both P_1 and P_2 send reports independent of their types. Therefore, in any equilibrium, no party can compute $y = x_1 \oplus x_2$ with better probability than guessing.

4.9.3 Recent Improvements

In [GK06], Gordon and Katz revisit the study of [HT04] of non-cooperative games. Their main contribution is a probabilistic two-party protocol for secret sharing with non-cooperative reconstruction — the protocol in [HT04] worked only for $n > 2$ parties, and it was even claimed in [HT04] that no two-party protocol existed. The two party protocol from [GK06] can be generalized to an arbitrary n parties and can then be used to implement non-cooperative evaluation of any function. The protocol of [GK06] is simpler and more efficient than the one of [HT04], and in addition has the advantage of eliminating an undesirable equilibrium present in the protocol of [HT04]. Gordon and Katz also show how to transform their solution so that the dealer of the secret has to play only in the distribution phase. In [HT04] the dealer had to be present continuously throughout the (non-fixed round) reconstruction phase.

An important property of all these results is that like [HT04] they require simultaneous broadcast but unlike [HT04] they do not need private channels. While the results of [GK06] are implied by the more general results of [LMPS04, LMS05, IML05], the protocols of [GK06] rely on weaker assumptions with respect to the model of communication (as [LMPS04, LMS05, IML05] need envelopes and ballot-boxes).

The Protocol. We now review how Gordon and Katz achieve two-party *rational secret sharing* (i.e., secret sharing with non-cooperative reconstruction). The basic protocol proceeds with the dealer being present throughout the reconstruction phase, which proceeds in a number of trials. At the beginning of each trial the dealer with probability β sends shares of the correct secret while with probability $1 - \beta$ it sends shares of a random string that is out of the space of the possible secrets. The work of the players is then very simply to broadcast the received shares and go to the next trial if all players correctly sent their shares and the reconstruction gives a string that is out of the space of the possible secrets. By carefully setting β on top of the players' utilities, Gordon and Katz prove that the above protocol constitutes a Nash equilibrium for t-out-of-n secret sharing and that this equilibrium is an IEDSS.

The above protocol works even for $n = 2$ parties, which were claimed impossible in [HT04]. The reason for this seeming paradox is that the proof in [HT04] only considers dealers which distribute shares of the correct secret in each trial, and for such a dealer the impossibility result is true. The model does however not restrict the dealer to such a behavior.

Extensions and Generalizations. An (inefficient) transformation of the protocol above produces a protocol where the dealer works only once, at the beginning of the protocol. The transformation is based on an execution per iteration of a secure computation protocol against one malicious player.

A generalization of the above techniques is then used for rational secure function evaluation under the restriction that all players are rational and their utilities are such that they prefer to learn the output. The basic idea is that on top of a function f, there exists a function f' that performs the same computation of f and moreover generates a secret sharing of the output. First this function is computed securely, and then the sharing is reconstructed using

the non-cooperative reconstruction. We leave out further details of the protocol, as it is similar to a protocol which we look at in Section 4.10.4.

4.9.4 Conclusion

The notion of implementation in [HT04] is that of a protocol implementing a special equilibrium. One could, however, try to compare a non-cooperative implementation π_f of a function f to the function mediator M_f, and ask whether replacing M_f by π_f in some context would preserve some properties. It is clear that π_f seldom would preserve all equilibria, as π_f allows much more signaling because the parties have access to a broadcast channel and there is no penalties associated to "misusing" the broadcast for signaling. One might ask whether replacing M_f by π_f in some context which is itself a non-cooperative context would preserve whether the context itself is a non-cooperative implementation of some function. This seems very unlikely, however, as the utilities associated to a sub-protocol of a non-cooperative implementation need not be of the non-cooperative type. Often there are not even well-defined utilities associated to a sub-protocol, only to the final outcome. Therefore it might not even make sense to ask whether the utilities of a sub-protocol are non-cooperative.

The work by Gordon and Katz explores the challenging direction of obtaining feasibility results under weaker assumptions. Moreover, they provide an efficient protocol for rational secret sharing only, thus leaving open the question of designing efficient protocols for other different games in presence of rational players.

4.10 Non-Cooperative Secret Sharing and Function Evaluation (Mixed Behavior)

In [LT06] Lysyanskaya and Triandopoulos consider a model closely related to the one considered in Section 4.9, but introducing parties corrupted in the Byzantine sense that they have no utility and behave arbitrarily. Formally these corrupted parties are essentially handled by universal quantification.[20]

The authors divide the parties into two sets, the *rational parties* and the *adversarial parties*. As in Section 4.9, it is assumed that rational parties behave rationally, in the game-theoretic sense, and that they prefer to learn the function value over not learning it and prefer as few as possible of the other parties to learn the value. The adversarial parties, of which there are t, are assumed to behave arbitrarily. This e.g. models t parties being infected by a virus. The goal is to construct a protocol from which the honest parties have no incentive to deviate, even when they know that some t of the other parties might behave arbitrarily.[21] In [LT06] the new model is called a *mixed-behavior model*.

When setting $t = 0$, the model proposed by [LT06] is essentially equivalent to the one from Section 4.9. So, having also to consider the cases $t > 0$ when proving security makes the model in [LT06] more strict. Indeed, the protocols proposed in [HT04] turn out *not* to be secure in the model of [LT06], and this happens already when $t \geq 3$.

To give an example of the additional difficulties met in the mixed-behavior model, we note that a protocol based on secret sharing with threshold d cannot tolerate t adversarial parties.

[20] As an example a mixed Nash equilibrium is a strategy which is a Nash equilibrium for all possible behaviors of the corrupted parties.

[21] Observe that this notion is somewhat similar to the notion of a dominating equilibrium.

To see this, observe that an adversary controlling t parties can approach a rational party and gives it the t adversarial parties' shares of all sharings. Now this rational party has $t+1$ shares of all sharings. So, if $d = t$, which is often the case in SFE, then the rational party will now know the value of all sharings, in particular the inputs of all parties. Therefore the rational party now has no incentive to keep participating in the protocol. Even more to the point, because of this problem, no rational party would have shared its input with threshold t in the first place. This shows that protocols based on secret sharing must use sharings with threshold $d > t$.

In [LT06] the authors manage to use sharings with threshold $d = t + 1$. At a closer look, this seems very surprising. Assume namely that an adversary hands the t adversarial shares to two rational parties P_1 and P_2.[22] Now P_1 and P_2 each knows $t + 1$ shares and therefore only need one more share to know all inputs. Since P_1 and P_2 together know $t+2$ shares of all inputs, it follows that they together have enough information to compute $f(x_1, \ldots, x_n)$. So, P_1 and P_2 could run a secure two-party computation between them to compute $f(x_1, \ldots, x_n)$. The reason why this is not considered a violation of the security notion introduced in [LT06] is that it would require both P_1 and P_2 to deviate from the protocol. If only P_1 deviates, and P_2 does not engage in the run of the two-party computation, then P_1 obtains nothing. Therefore the protocol is a Nash equilibrium, as being a Nash equilibrium only requires that there is no strategy which P_1 on its own could adopt to increase its utility (in this case, learn $f(x_1, \ldots, x_n)$ without e.g. P_3 learning it.) If on the other hand the equilibrium was to be stable against collusions of size 2 between rational parties, then secret sharing with threshold $t + 2$ should be used to combat the above possible collusion between P_1 and P_2.

As in [HT04] the authors in [LT06] make assumptions on the utilities of the rational parties and implement only a limited class of functions. Because they introduced t adversarial parties, however, they have to further restrict the class of functions being implemented. For this purpose they introduce the class of t-NCC functions, which is essentially the functions from NCC with the extra property that the output of the function can never be computed from $t + 1$ inputs alone.

Below we sketch the definitional approach from [LT06], going over the assumptions on utilities (Section 4.10.1), the notion of a mixed Nash equilibrium (Section 4.10.2) and t-NCC functions (Section 4.10.3) and culminating in the notion of non-cooperatively computable functions for the mixed-behavior model (Section 4.10.5). We will also sketch the protocol from [LT06] (Section 4.10.4).

4.10.1 Assumptions on Utilities

As in [HT04] the utility is computed from the parties' ability to compute certain information. Specifically a function $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ is given, where x_i is an input of P_i and y_i is the value that P_i is interested in. It is assumed that the utility of P_i is given by its ability to compute y_i, which is essentially boiled down to the probability p_i that the private output of P_i is y_i. Since the authors in [LT06] need a definition where the case $p_i = 1$ receives a special attention, they allow the utilities to depend on p_i. In more detail, for a given input distribution $(x_1, \ldots, x_n) \leftarrow X$ and a rational party P_i we let Y_i denote the private output of P_i in $\pi(X)$, and we assume wlog that Y_i includes all messages received by P_i during the protocol run. We then let p_i denote the probability of guessing y_i given Y_i and we let q_i denote

[22] To convince them that he did so, he could multicast the shares to the two parties (allowing both to see that both got them), along with a proof that the shares are correct.

the probability of guessing y_i given only x_i (in both cases using the best guessing algorithm, which might depend on π and X). We then let $\mu_i = (p_i - q_i)(1 - q_i)$, assuming that $q_i < 1$. Since $p_i \geq q_i$ and $p_i \leq 1$ we have that $\mu_i \in [0, 1]$ is a measure of how much the protocol π helps P_i to better compute y_i. For the adversarial parties P_j we let $\mu_j = \perp$ for some reserved symbol $\perp$. In [LT06] the following definitions are made on the utilities:

utility structure: The utilities of $\pi(X)$ are computed from $\vec{p} = (\mu_1, \ldots, \mu_n)$. Below, let $\vec{p} = (\mu_1, \ldots, \mu_n)$ and let $\vec{p'} = (\mu'_1, \ldots, \mu'_n)$.

P_i **wants to learn:** If there exists a *rational* P_i such that $\mu_i > \mu'_i$ and $\mu_j = \mu'_j$ for all *rational* $P_j \neq P_i$, then $u_i(\vec{\mu}) > u_i(\vec{\mu'})$.

P_i **does not want others to learn:** If there exists a *rational* P_j such that $\mu_j > \mu'_j$ and $\mu_i = \mu'_i$ for all *rational* $P_i \neq P_j$, then $u_i(\vec{\mu}) < u_i(\vec{\mu'})$ for all *rational* $P_i \neq P_j$.

worst-case outcome: If there exists a *rational* P_j such that $\mu_j = 1$, then $u_i(\vec{\mu}) = 0$ for all *rational* P_i where $\mu_i < 1$.[23]

no adversarial utility: For all adversarial parties P_i we have that $u_i = 0$.

The first three assumptions essentially state that all parties get utility from an interaction allowing them to better guess their desired outcome and get utility from an interaction making it harder for the other parties to guess their desired outcome. The assumption **worst-case outcome** is not motivated in [LT06] and seems to be there simply to allow to prove the protocol secure.

4.10.2 Mixed Nash Equilibrium

We present the notion from [LT06] of a mixed Nash equilibrium.

The adversarial parties are given by a subset $A \subset \{1, \ldots, n\}$. The adversary is then an alternative program for each adversarial party, $\mathcal{A} = \{\tilde{A}_i\}_{i \in A}$. For a given protocol $\pi = (A_1, \ldots, A_n)$ we can then define $\pi^{\mathcal{A}} = (\mathcal{A}, \pi_{-A})$ to be the protocol $(\hat{A}_1, \ldots, \hat{A}_n)$, where $\hat{A}_i = \tilde{A}_i$ for the *adversarial parties* $i \in A$ and $\hat{A}_i = A_i$ for the *rational parties* $i \notin A$.

A protocol π is then called a mixed Nash equilibrium for a game Game $= (\mathcal{R}, \Omega, u, X)$ ([LT06, Definition 6]) with adversarial parties $A \subseteq \{1, \ldots, n\}$, if for all rational parties $i \notin A$ and all programs $\tilde{A}_i$ and all adversaries $\mathcal{A} = \{\tilde{A}_i\}_{i \in A}$ it holds that $u_i(\pi^{\mathcal{A}}(X)) \leq u_i(\pi^{\mathcal{A}'}(X))$, where $\pi^{\mathcal{A}'} = (\tilde{A}_i, \pi^{\mathcal{A}}_{-i})$.[24]

We can actually simplify this definition slightly. Note that the universal quantifications over i, $\tilde{A}_i$ and $\mathcal{A}$ occur at the same place, meaning that we can change the order to: "for all adversaries $\mathcal{A} = \{\tilde{A}_i\}_{i \in A}$, for all rational parties $P_i \notin A$ and for all programs $\tilde{A}_i$ it holds that $u_i(\pi^{\mathcal{A}}(X)) \leq u_i(\pi^{\mathcal{A}'}(X))$". But, this is exactly the same as requiring that $\pi^{\mathcal{A}}$ is a Nash equilibrium for all $\mathcal{A}$. So, we have the following simpler definition.

Definition 4.7 (Mixed Nash Equilibrium) *Let* Game $= (\mathcal{R}, \Omega, u, X)$ *be any game, and let* $A \subseteq \{P_1, \ldots, P_n\}$. *Let* $\pi = (A_1, \ldots, A_n)$ *be a protocol for* Game. *We say that* π *is a* t-mixed Nash equilibrium *for* Game *and adversarial parties* $A \subseteq \{1, \ldots, n\}$ *if it holds for all* $\mathcal{A} = \{\tilde{A}_i\}_{i \in A}$ *that* $\pi_{\mathcal{A}}$ *is a Nash equilibrium for* Game.

[23] It is assumed that $u_i \geq 0$ for all rational P_i, so that 0 is the lowest possible utility.

[24] Note that there is a typo in [LT06, Definition 6], where σ_i on the right-hand-side should be σ_i^*.

4.10.3 t-NCC Function

The t-NCC functions are essentially defined to be the class of functions f which can be computed non-cooperatively by n parties given a black-box for evaluating f, when there are t adversarial parties. More specifically, a function f is called a t-NCC function for an input distribution X and utility u if the following holds. Let $A \subset \{1, \ldots, n\}$ quantify over all A with $|A| = t$. Let $(x_1, \ldots, x_n) \leftarrow X$ be the distribution on the inputs, let $\mathcal{A}$ be an adversary which defines alternative inputs $x_A = \{x'_i\}_{i \in A}$ for the adversarial parties, and let $x' = (x'_1, \ldots, x'_n) = \mathcal{A}(x)$ be an input distribution defined by letting x'_i be the alternative input defined by $\mathcal{A}$ for $i \in A$ and letting $x'_i = x_i$ for $i \notin A$. The function f is *non-cooperatively computable* for X and u if:

1. The probability of guessing $y_i = f(x_1, \ldots, x_n)$ given (x_i, x_A) is equal to the probability q_i of guessing y_i given x_i.
2. The output of the function f cannot be changed by only changing the input of the adversarial parties. I.e., it holds for all adversaries $\mathcal{A}$ that when $x \leftarrow X$ and $x' = \mathcal{A}(x)$, then $f_i(x) = f_i(x')$ for all $i \notin A$.
3. When running with the resource $\mathcal{R}_f$ which on input $(x_1, \ldots, x_n)$ outputs $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, the protocol where all P_i input x_i to $\mathcal{R}_f$ is a mixed Nash equilibrium.[25]

Note that the class of t-NCC functions is very special. E.g., replacing any t inputs does not change the output, essentially making the input distribution an error correcting code tolerating t mistakes and making f the procedure for correcting the mistakes.

4.10.4 The Protocol

To get a feeling for Definition 4.7 it is instructive to consider a brief sketch of the protocol from [LT06]. We focus on the case $(y, \ldots, y) = f(x_1, \ldots, x_n)$ where all parties have the same output.

1. Each party P_i has a public key pk_i for an encryption scheme.
2. First the parties run a generic SFE protocol (see Chapter 3) tolerating n corrupted parties, computing a common output

$$(C_1, \ldots, C_n) = (E_{pk_1}(y_1), \ldots, E_{pk_n}(y_n)) \ ,$$

   where $(y_1, \ldots, y_n)$ is a uniformly random n-out-of-m secret sharing[26] of $y = f'(x_1, \ldots, x_n)$, where $m = t+1$ and $f'(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ with probability $\frac{1}{2}$ and $f'(x_1, \ldots, x_n) = \perp$ otherwise, where $\perp$ is some reserved symbol signaling that no output was computed. If during this computation any party is seen deviating from the protocol, then all parties stop the protocol, yielding no output.
3. After computing $(C_1, \ldots, C_n)$, each P_i simultaneously broadcasts y_i, along with a proof that C_i is an encryption of y_i. There are now three cases:

[25] In [LT06] some more technical details are required, but we have focused on the essence of the definition. As an example [LT06] requires that submitting any $x'_i \neq x_i$ gives strictly less utility. For the details of the definitions, see [LT06].

[26] I.e., a secret sharing between n parties with threshold m.

successful, result: All n parties broadcast y_i. In this case all parties compute y and if $y \neq \perp$, then all parties output y.

successful, no result: All n parties broadcast y_i, but $y = \perp$. In this case the computation is run again from Step 2, using fresh randomness.

unsuccessful: Less than n parties broadcast y_i. In this case all parties which have at least $m + 1 = t + 2$ shares compute y. If $y \neq \perp$, then they output y. Otherwise no output is computed.

This protocol can be seen to be a mixed Nash equilibrium. We consider a rational party P_i and argue that P_i has no incentive to deviate. Clearly P_i has now incentive to deviate in Step 2 as no party gains any new information during this step. At the beginning of Step 3 the parties then hold a n-out-of-$t + 1$ sharing of y, where $y = f(x_1, \ldots, x_n)$ or $y = \perp$.[27] The party P_i can now choose to broadcast y_i or not broadcast y_i. By setting t such that $n \geq 2t+3$ it is ensured that there are $(n - t) - 1 \geq t + 2$ rational parties besides P_i, and these $t + 2$ parties will broadcast their shares when they follow the protocol. So, no matter whether P_i broadcasts its share or not, if $y = f(x_1, \ldots, x_n)$, then all parties learn $y = f(x_1, \ldots, x_n)$. If on the other hand $y = \perp$ and P_i does not broadcast its share, then the computation halts and no party learns the output. Therefore a rational P_i will prefer to broadcast its share. The reason why a n-out-of-$t + 1$ sharing is used is that no honest party which receives the shares of the adversarial parties should have enough information to learn y, which would namely give it an incentive to not broadcasting its share.

4.10.5 Non-Cooperatively Computable (Mixed)

We are then ready to state what it means for a protocol to non-cooperatively compute f in the t-mixed-behavior model.

Definition 4.8 *A protocol π* non-cooperatively computes *f* in the t-mixed-behavior model *(on input distribution X and with utilities u fulfilling the requirements in Section 4.10.1) if:*

Correctness and security: *The protocol π is a t-secure SFE protocol for f.*

Nash equilibrium: *The protocol π is a t-mixed Nash equilibrium for $(\mathcal{R}, X, u)$.*[28]

Survival condition: *The protocol π is an IEDSS for $(\mathcal{R}, X, u)$.*

More details on this definition can be found in [LT06]; E.g., the notion of IEDSS has to be appropriately extended to the mixed-behavior model.

In Section 4.10.4 we already sketched the protocol from [LT06] and argued that it is a t-mixed Nash equilibrium. That it is a t-secure SFE protocol in the cryptographic sense is fairly straightforward. That the protocol is an IEDSS was touched upon already in Section 4.10.4: At all points in the protocol, if a party P_i deviates from the protocol it sometimes excludes P_i from learning the outcome. Therefore a strategy A'_i for P_i which deviates from the strategy A_i specified in Section 4.10.4 will do worse than A_i in some context[29] and therefore cannot

[27] Note that indeed $y = f(x_1, \ldots, x_n)$ when $y \neq \perp$, as the t adversarial parties replacing their inputs cannot change the output of f.

[28] We omit Ω in the notation of the game as there is no common output in the game for computing f; The utilities are given by the private output only.

[29] In particular when all other parties are running according to the protocol from Section 4.10.4.

weakly dominate A_i. That this argument holds also for iterated elimination is proved in detail in [LT06].

4.10.6 Computational Variants

In [LT06] all definitions are given both in information theoretic variants and in computational variants. In the above we focused on the information theoretic variants for clarity. Here we briefly discuss how computational variants are derived.

Most of the details in defining computational variants is fairly straightforward, and follows the lines from Section 4.2.9. The main obstacle is to define the probabilities q_i and p_i, where q_i is the probability of computing $y_i = f(x_1, \ldots, x_n)$ given x_i and p_i is the probability of computing $y_i = f(x_1, \ldots, x_n)$ given x_i and the view Y_i of P_i in the protocol. In [LT06] these probabilities are defined by the probability that $F(x_i) = y_i$ and $F(Y_i) = y_i$ for the best PPT algorithm F.

4.10.7 Conclusion

We draw a few conclusions about the work from [LT06].

IEDSS versus Domination

In [LT06] a preferred protocol for the t-mixed-behavior model is defined to be a protocol which is a t-secure SFE protocol in the usual sense, and which is in addition a Nash equilibrium no matter how the t adversarial parties behave. Finally, it is required that the protocol is an IEDSS, to refine the notion of Nash equilibrium. The reason why IEDSS is used for refinement, as opposed to requiring that the protocol is a weakly dominating strategy, is that it seems impossible to find dominating strategies for as complex settings as the t-mixed-behavior model, at least when realistic communication resources are used. As an example, consider the protocol from Section 4.10.4. This protocol is an IEDSS, but is not a weakly dominating strategy. To see this, consider the following protocol: After the shares of y have been computed, some collusion of $t+2$ of the parties use a side-channel to run an SFE protocol on their shares to compute whether or not $y = \perp$. If $y = \perp$, then they participate honestly in the protocol. If $y \neq \perp$, then these $t+2$ parties first participate in the simultaneous broadcast by *not* broadcasting their shares. Then, if each of the $t+2$ parties did *not* broadcast their shares and the remaining parties *did* broadcast their shares, then the $t+2$ colluding parties exchange their $t+2$ shares and reconstruct the output $y = f(x_1, \ldots, x_n)$. This is an example of a protocol where it is a disadvantage for the parties to broadcast their share of y, meaning that the proposed strategy from Section 4.10.4 where the share is always broadcast is not dominating.

Non-Cooperative Computation Beyond NCC

In [LT06] all definitions rely heavily on the fact that only t-NCC functions are considered. It would however be interesting to extend the definitions beyond that of t-NCC functions. Naïvely it seems natural to consider only t-NCC, as for functions f which are not t-NCC we cannot make the honest parties give their correct input even in the ideal setting where the parties have access to an ideal functionality for computing f. At a closer look, the restriction to functions in t-NCC is, however, unnatural, as we can still require a protocol to be at

least as secure as the ideal setting. This a definitional approach from cryptographic protocol theory where an implementation is called secure if it is as secure as the ideal protocol. From a cryptographic point of view it would therefore be natural to attempt to extend the approach in [HT04, GK06, LT06] to the set of all functions by saying that a protocol π is a (t-)secure non-cooperative implementation of f if the implementation is at least as secure as the natural protocol for the ideal model; If not even the ideal model forces rational parties to give the correct input, then of course the implementation cannot force this either. However, it makes sense to consider protocols which then have no other flaw than this unavoidable flaw.

Computational Variants

As part of the computational variants of the definitions in [LT06] the authors defined the probability that $F(x_i) = y_i$ and $F(Y_i) = y_i$ for the *best* PPT algorithm F. Formally this seems to not make sense. Recall e.g. that at some point in the protocol from Section 4.10.4 the shares of y are made public. This means that after this point in the protocol, if $y = f(x_1, \ldots, x_n)$, then the views of all parties contain $y = f(x_1, \ldots, x_n)$ in an information theoretic sense. So, any party P_i which does not know y can always invest a little more time in trying to break the encryption scheme to get the shares and compute y. Therefore there does not exist a best PPT algorithm F; For any algorithm running in time k^c there will exist a better one running in time k^d for $d > c$, assuming that the encryption scheme is secure against all poly-time algorithms. To deal with this formal problem several approaches could be taken. First of all, one could consider concrete time, allowing each P_i (plus F) to use some fixed number of steps. This would give a well-defined q_i and p_i, but might give a much more involved analysis. Secondly, one could remove the use of F and require that the code A_i of P_i outputs some y_i' and then define p_i by the probability that $y_i' = y_i$. Then p_i would be defined by A_i in a natural way. How to define q_i in a natural way is however less obvious.

4.11 Further Reading

The above chapter is by far an exhaustive survey of the research on rational cryptography, but has focused on some of the defining work. A list of other important work is mentioned now, for further reading. In [ADGH06] Abraham *et al.* study rational secret sharing and multiparty computation. In [EL04] Elkind and Lipmaa consider online auctions from a rational cryptography point of view. In [LMS05] Lepinski, Micali and Shelat study collusion-free computation.

Chapter 5

Cryptographic Protocol Theory Applied to Economy

In this chapter we survey some economic problems which have or could benefit from the use of techniques from cryptographic protocol theory. We focus on auctions, fair division collaborative benchmarking and forecasting, and polling.

5.1 Auctions

Auctions are simple and robust mechanisms for selling nonstandard and short supply items with uncertain market values. Auction mechanisms discover the optimal price for a commodity through the bidding action of self-interested agents [UPF98]. In an auction scheme there can be multiple sellers and one buyer or many buyers (bidders) and one seller (auctioneer), or both.

5.1.1 Introduction to Auction Mechanisms

The auction mechanism expresses terms and conditions for determining the winner and the price that it has to pay for the item.

We characterize *price discovery* mechanisms as progressive (ascending price, descending price) or sealed bid, and the price can be the highest price or second-highest one.

With respect to the privacy of the bids, we can identify *sealed-bid* auctions where all bids are private, or *open-cry* bids where bidders publish their valuations.

While there are many combinations of auction characteristics, some auction mechanisms are redundant and do not have general use, as noted in [UPF98, PUF99]. For example the second-price ascending auction is strategically equivalent to the first-price ascending auction for small bid increments, and will have the same revenue and efficiency properties.

English Auction. The *English auction* is the most common known auction type where each bidder offers a price for a good against one another by offering a higher price than the previous one step by step. The auctioneer continuously reveals the highest bid received, and the asked price is a minimum increment above the price of the current highest bid. The auction finally ends when a bidder offers the highest price and no more participants are willing to offer more. The winner finally gets the good paying his (the highest) bid. A variation of English auction

is *Japanese auction* where the price increases gradually while participants quit the auction one after another by showing their disagreement with the price.

Dutch Auction. In a *Dutch auction*, an auction manager (an auctioneer or a seller) begins with a high asking price and reduces the price until one bidder accepts the price for the first time. The auction is immediately terminated.

1^{st}-Price Sealed-Bid Auction. In a *first-price sealed-bid auction* each bidder secretly and independently from others submits a single bid to the auction manager only once, and the bidder who offers the highest price gets the good by paying the amount specified in his bid after a fixed period of time.

2^{nd}-Price Sealed-Bid Auction (Vickrey). The *Vickrey* or *second-price sealed-bid auction* was proposed by Nobel Prize laureate William S. Vickrey in 1961 [Vic61]. This scheme is identical to the 1^{st}-price sealed-bid auction, except that the winning bidder has to pay the amount of the second-highest bid.

The Vickrey auction is an improvement of the 1^{st}-price sealed-bid auction from an economic viewpoint, as it has a dominating strategy for bidding (see Chapter 4). This can be seen as follows, citing [BW01]: "... applying this strategy the bidder receives the highest possible payoff, no matter what strategies are used by the other bidders. The dominating strategy is to bid one's true valuation of a good. Even if an agent knows all the other bids in advance, he still does best by bidding his private value."

Contingent Bid Auctions. In this auction mechanism there is a seller who has some items for sale as usual, and many bidders who bid for those items. The distinction of this mechanism is that the buyers are uncertain about the quantity they will need at a later point in time. Besides, the uncertainty information remains private for each of the buyers, and the seller does not have any clue whether this demand will be used by the buyers. The seller then has to decide who is the winner, trying to maximize his revenue, based on the information available.

Further details on contigent-bid auction and their variations are presented in [Han85, RKV00, KS03, BS04, MCW05].

Combinatorial Auctions. A *combinatorial bid* is a bid in a multi-unit auction (more than one item is available for sale) in which a bidder offers a price for a collection of goods (of the bidder's choosing) rather than placing a bid on each item separately. In a *combinatorial auction*, the auctioneer selects a set of these combinatorial bids which yields the maximal revenue without assigning any object to more than one bidder. In other words, combinatorial auctions are those auctions in which bidders can place bids on combinations of items, called "packages", rather than just individual items. The study of combinatorial auctions is inherently interdisciplinary.

Combinatorial auctions are in the first place auctions, a topic that economists have extensively studied. Package bidding brings in operations research, especially techniques from combinatorial optimization and mathematical programming. Finally, computer science is concerned with the expressiveness of various bidding languages, and the algorithmic aspects of the combinatorial problem. The study of combinatorial auctions thus lies at the intersection

of economics, operations research, and computer science. The book by Cramton, Shoham and Steinberg [CSS06] looks at combinatorial auctions from all three perspectives, and is mainly a compilation of various papers in the area of combinatorial auction.

5.1.2 Cryptographic Protocols for Auctions

Given an auction mechanism the straight-forward approach to employ it on for instance the Internet is to implement a mediator for the mechanism, using, e.g., an SFE protocol, or even better, one of the rationally secure protocols from Chapter 4. The basic approach of building a Boolean circuit that computes the auction outcome on binary representations of bids and then applying a general scheme is, however, not feasible, as these general schemes are quite inefficient [Bra03b, NPS99] and the circuit depth, and/or the round complexity, depends on the number of bidders and the bid size.

For this reason people have developed special purpose auction protocols with variants of goals, security levels, trust models, level of robustness, etc. Such a protocol must assure the requirements (price discovery mechanism, winner determination, privacy of bids if corresponds, etc) along with some other correctness concerns or efficiency considerations. In the following sections we survey some general implementation considerations, some recent auction protocols and some of the techniques which are used in the design of auction protocol.

5.1.3 General Implementation Considerations

We first discuss some general issues associated to implementing a mechanism.

Trust Models

In general the bidders have to trust the auctioneer when he declares the winner and the selling price according to the mechanism used. Even the bidders themselves can mistrust each other, which can be natural for the existing competition. Often, this trust cannot be guaranteed and therefore, based on such mistrust, some procedures have to achieve the correct result.

In that direction there are different approaches. One consists in letting the bidders compute the auction results by interaction between them. This corresponds to a *bidder-solved* protocol [Bra03b]. Other approaches introduce third-party agents to the scene. The overall goal is to put a high level of trust to these parties in order to reduce the complexity of the protocol, for example, a bidder-resolved auction protocol. Sometimes these agents are assumed to have some good reputation which can be lost in case of suspicion or provable misbehavior. Or they cannot be successful when they misbehave unless a collusion of a given number of them is formed.

Most of the cryptographic auction schemes work in one of the following two trust models [LAN02]:

The threshold (symmetric) trust model: The threshold (symmetric) trust model where the tasks of the seller are executed by $N > 1$ servers, with less than $N/3$ (or $N/2$ depending on the precise cryptographic model) servers assumed to be dishonest.

The two-party (asymmetric) model: The two-party (asymmetric) model with a seller or auctioneer and an auction authority or issuer, where at least one of them is assumed to be honest. In this model, both seller and authority are usually assigned complementary duties and are supposed to verify the actions of each other.

General Requirements

In the following, we list some of the possible requirements that are desirable to certain implementations, which of course depend on various parameters, like the mechanism and other aspects under consideration [Omo02].

Bidder privacy: If anonymity for an authority is not realized in an electronic auction, much information of who wants a good and a bidder's history of bidding is easily stored. Such information is valuable and can be easily reached and selected. It may be bought and sold through illegal channels. For example, once a bidder participates in the old style stamps or coins auction, many direct mails about different cosmetics may be sent to the participants. Therefore, nobody should be able to reach personal data from an auction. In general, privacy relies neither on trusted third parties (like auctioneers) or trusted fractions of bidders, nor on computational intractability assumptions (like the hardness of factoring).

Correctness of the system: A bidder is of course anxious about the correctness of the system. Nobody should be able to impersonate a certain bidder and all bids should be fairly dealt with. On the other hand, an auctioneer has also to be assured that no bidder can cheat in the auction scheme.

Efficiency: A bidder wants to quickly place a bid in a simple way in an auction system. Also, it is unpleasant that computing the results of auction takes too much time. The

following properties are desirable for an electronic auction scheme:

Anonymity: If required by the auction mechanism, nobody, including an authority, should identify bidders (probably losers) even after the opening phase.

Non-cancelability: A winner cannot deny that he submitted the highest bid after the winner decision procedure. A winner is exactly identified.

Public verifiability: Anybody can publicly verify that a winning bid is the highest value of all bids and publicly confirm whether a winner is valid or not.

Unforgeability: Nobody can pretend to be a certain bidder.

Robustness: Even if a malicious bidder sends an invalid bid, the auction process is not affected. In particular this robustness is said to be *strong* when malicious bidders are discarded and the remaining participants can compute the outcome of the auction protocol over all the bids. On the other hand, the robustness is *weak* when the remaining parties can only compute the result of the auction based on their inputs (better if they don't restart).

Fairness: Any participant quiting the protocol does not get any advantage from quitting. Meaning that, after a participant learns a piece information in the protocol he cannot prevent others to learn the information directed for them.

Revocation: A bidder can withdraw his bid without affecting the auction process. This can be an opposite requirement to strong robustness.

One-time registration: Any bidder can participate in plural auctions by only one-time registration. Even if the bidder is the winner for an auction, he can participate in the next auction without registering again, maintaining the anonymity for the authorities.

Unlinkability among the plural auctions: Nobody can link the same bidder's bids among the plural auctions.

Independent authority's powers: There is no single authority that can break the anonymity and secrecy of bids.

Linkability in an auction: Anybody can link all the bids a bidder places and how many times has a bidder placed bids in an auction.

Secrecy of highest bid: The protocol should not disclose the higher bidding prices except the winning price to be paid. This property is valid for the second price (Vickrey) sealed bid and $(M + 1)$-st auctions. This property protects the winner's privacy.

Bid-Rigging and Receipt-Freeness

Bid-rigging is an illegal agreement between two or more competitors. It is a dangerous attack in electronic auctions. If the bid-rigging happens in an online auction, the winner can get the bidding item with an unreasonably low price. There are some very common bid-rigging practices [DoJ00]:

Bid suppression: In bid suppression schemes, one or more competitors who otherwise would be expected to bid, agree to refrain from bidding or withdraw a previously submitted bid so that the designated winning competitor's bid will be accepted.

Complementary bidding: Complementary bidding (also called "cover" or "courtesy" bidding) occurs when one or more competitors agree(s) to submit bids that are either too high to be accepted or contain special conditions that will not be acceptable to the purchaser. Such bids are not intended to secure the purchaser's acceptance, but are merely designed to give the appearance of genuine competitive bidding. Complementary bidding schemes are the most frequently occurring forms of bid rigging and they defraud purchasers by creating the appearance of competition to conceal secretly inflated prices.

Bid rotation: Bid rotation schemes are similar to complementary bidding schemes. In bid rotation schemes, all conspirators submit bids, but take turns on being the low bidder. The terms of the rotation may vary; for example competitors may take turns on contracts according to the size of the contract, allocating equal amounts to each conspirator or allocating volumes that correspond to the size of each conspirator company. A strict bid rotation pattern defies the law of chance and suggests collusion is taking place.

Subcontract bid-rigging: Subcontracting arrangements are often part of a bid-rigging scheme. Competitors agree not to bid or to submit a losing bid in exchange for subcontracts or supply contracts from the successful low bidder. In some schemes the collusion may take place in the bidding process; for example, a low bidder may agree to withdraw its winning bid in favor of the next low bidder in exchange for a lucrative subcontract that divides the illegally obtained higher price between them.

These forms of bid-rigging are not mutually exclusive of one another, and two or more of these practices could occur at the same time. For example, if one member of the bidding ring is designated to win a particular contract, his co-conspirators could avoid winning either by not bidding ("bid suppression"), or by submitting a high bid ("cover bidding").

Bid-rigging is a form of fraud, and almost always results in economic harm to the agency which is seeking the bids, and to the public, who ultimately bears the costs as taxpayers or consumers. The paper by Abe and Suzuki [AS02b] points out the risk of bid-rigging in electronic sealed-bid auctions and provides a receipt-free sealed-bid auction scheme to prevent it. They claim that to prevent bid-rigging and achieve a fair auction, the sealed-bid auction must satisfy secrecy of bidding price, public verifiability and receipt-freeness. Receipt-freeness means that anyone, even the bidder himself, must not be able to prove any information about the bidding price except what can be found from the result of the auction. If a bidder can prove his bidding price, a coercer/buyer can perform bid-rigging as described above.

5.1.4 Recent Auction Protocols and Techniques

The paper [Cac99] describes a protocol for the following problem: A wants to buy some good from B if the price is less than a. B would like to sell, but only for more than b, and neither of them wants to reveal the secret bounds. Will the deal take place? The solution uses an oblivious third party T who learns no information about a or b, not even whether $a > b$. The protocol needs only a single round of interaction, ensures fairness, and is not based on general SFE techniques. It uses a construction which combines homomorphic encryption with the hiding assumption. Applications include bargaining between two parties and secure and efficient auctions in the absence of a fully trusted auction service.

Starting from the framework put forth by Cramer, Damgård and Nielsen [CDN01] based on threshold homomorphic cryptosystems there are many building blocks that aim for generic constructions that can be used for implementing auction schemes. For the basic constructions the main approach consists in building an arithmetic circuit that will be evaluated for the participants to obtain the mediator's outcome. The evaluation requires interaction between participants when multiplication gates are considered, the rest of evaluation is for free, meaning that they can be solved by the participants themselves. Some of these works are: [ST04, DFK$^+$06, GSV07]. The special ingredient is their generality: the inputs as well as the output are encrypted and even unknown to the parties. The useful functionalities in the context of auctions can be, given the encryptions (in shared form) of x and y, compute an encryption of $[x > y]$ or $[x = y]$.[1]

Yao's garbled circuit [Yao82, Yao86] can also be used as building block for some particular problems in designing cryptographic solutions for auctions. This construction is general and can be applied for the evaluation of any computable function. For example, the protocol in [NPS99] defines the mechanism and the trust model, and then applies Yao's garbled circuit techniques to determine the outcome of the auction.

Brandt [Bra03b] proposes a constant-round protocol for the Vickrey auction mechanism which preserves the requirement of full-privacy without requiring any third-party.

Jakobsson and Juels [JJ00] propose an auction protocol based on a so-called mix-net protocol which takes as inputs a list of encryptions under a homomorphic scheme and produces another list with the same number of encryptions which is guaranteed to contain the same

[1] $[b]$ for a predicate b is 1 if b is true and 0 otherwise.

plaintext set with the same appearance frequency. With this building block they present a first-price sealed-bid auction protocol, using a threshold trust model along multiple servers. It is non-interactive in the sense that bidders do not need to interact after submitting their bids, is adaptable on other sealed-bid mechanism with low overhead, achieves full-privacy, robustness, public verifiability and the trust on servers can be adaptable.

The protocol by Lipmaa *et al.* [LAN02] requires a single semi-trusted third-party which is the auction authority. Bidders encrypt their bids using the auction authority's public key and send them to the seller who checks accompanying signatures, sorts the encrypted bids according to a pre-determined scheme (e.g. in lexicographic ciphertext order), and broadcasts them. The auction authority then opens all the bids, determines the selling price (e.g. the second highest bid), sends it to the seller, and proves its correctness by applying a sophisticated zero-knowledge proof. Winning bidders are required to claim that they won. The protocol is very efficient, but only provides limited privacy as the selling price is published and the auction authority learns all the bids. The only information hidden from the authority is the connection between bidders and bids. The scheme easily scales to a high number of bidders. However, the number of possible bids is severely limited. Neither the seller nor the auction authority can manipulate the outcome without being detected. A collusion of both instances uncovers complete information.

Abe and Suzuki's scheme[AS02a] is similar to the one in [Bra03b], using mix-and-match methods [JJ00] and in a trust model with two third-parties: the auctioneer and a trusted authority (which can be distributed to achieve threshold security). Bidders encrypt bidding vectors with a public key of the trusted authority, send them to the auctioneer, and prove their correctness. The auctioneer computes the sum of integrated bid vectors based on the homomorphic property of the cryptosystem. The resulting vector's components denote how many bidders are willing to pay a given price. Following this, the position of the 2nd-highest bid is determined by binary searching the lowest price that exactly one bidder is willing to pay. This is achieved by gradually releasing vector components to the authority who decrypts them and proves in zero-knowledge (using mix-and-match) that there were either more than one bidder or less than 2 bidders willing to pay. The whole process takes a logarithmic number of rounds in the number of possible bids. The authority learns statistical information during this process. The entire protocol is publicly verifiable and thus achieves robustness. Apparently, a collusion of the auctioneer and the trusted authority can learn complete information. Furthermore, the selling price must be publicly announced to convince losing bidders of their failure.

In [Bra02] Brandt proposes a verifiable bidder-resolved protocol for $(M+1)$st-price sealed-bid auctions in which only the winning bidders and the seller learn the selling price. It is based on verifiable secret sharing. In [Bra03a] the result is improved, by achieving full privacy using homomorphic ElGamal encryption and a private key that is distributed among the set of bidders. Bidders jointly compute the auction outcome on their own without uncovering any additional information in a constant number of rounds (three in the random oracle model).

In [AS02b] Abe and Suzuki propose a protocol with receipt-freeness. The paper [CBK03] then points out that Abe and Suzuki's scheme only provides receipt-freeness for losing bidders. This paper argues that it is more important to provide receipt-freeness for winners and proposes a new receipt-free sealed bid auction scheme using the homomorphic encryption technique. In contrast to Abe and Suzuki's scheme, they claim that this scheme satisfies privacy, correctness, public verifiability and receipt-freeness for all bidders. Also, this scheme is not based on the threshold trust model but a three party trust model, so it is more suitable

for real-life auctions.

The paper [HIS05] points out that the above-mentioned receipt-free schemes do not prevent perfectly the bid-rigging attack (including final receipt-free auction). The bid-collusion means that all the bidders have a prior consultation on a winner and a winning price with an unreasonably low price. Therefore, all the bidders know the estimated winner and winning price. They can check whether their promise is kept or not. If their promise is kept, the winner rewards for other bidders. Otherwise, the winner is punished by other bidders. To prevent the bid-collusion attack the bidder should not prove whether he cast a bid or not. They indicate that a bid-collision is a possible cause of an online auction. In the strict sense, the bid-collusion is different from the bid-rigging. The bid-collusion means that all the bidders have a prior consultation on a winner and a winning price with an unreasonably low price.

Brandt [Bra03a] investigates how to obtain bid privacy in sealed-bid auctions, focusing on unconditional full privacy. He shows that the first-price sealed-bid auction can be emulated by an unconditionally fully private protocol for which the round complexity is exponential in the number of bits that represent a bid. He also shows the impossibility of fully privately emulating the Vickrey auction for more than two bidders.

The scheme by Naor *et al.* [NPS99] is based on two servers (the auctioneer and the "auction issuer") and is applicable to general secure Boolean two-party computation for computing an $(M + 1)$st-price auction. The auction issuer generates a "garbled" Boolean circuit that computes the auction outcome for any given set of bids. The bidders then submit their encrypted bids. The auction issuer produces garbled inputs to the circuit from the bids and sends them to the auctioneer who then evaluates the circuit and publishes the result. The basic scheme is very efficient as it works on binary representations of bids. However, the auctioneer needs to verify that the auction issuer's garbled circuit is correct. This is solved by the "cut-and-choose" technique which guarantees correctness with exponentially small error probability. Although this scheme is based on a sophisticated trust model, privacy vanishes if the auction issuer and the auctioneer collude.

In [Bra03b], two security concerns of the Vickrey auction are presented: the vulnerability to a lying auctioneer and the reluctance of bidders to reveal their private valuations. Also, a technique that allows to securely perform second-price auctions is described. This is achieved using the announcement of encrypted binary bidding lists on a blackboard.

In [BDJ+06] the main contributions are: A generic setup for secure evaluation of integer arithmetic including comparisons; general double auctions expressed by such operations; a real world double auction tailored to the complexity and performance of the basic primitives $+$ and $\leq$ and finally evidence that the approach is practically feasible based on experiments with prototypes.

5.1.5 Mechanism Design and Cryptography

The classical approach in designing cryptographic protocols which implement a given auction mechanism is to put cryptography on top of the mechanism and then, as usual, prove the security properties that the cryptographic protocol enjoys. In general such security properties are related to the security against sellers and bid privacy of the on-line auctions. Many authors have been proposing a wide range of cryptographic schemes that guarantee security against sellers and privacy of various auction mechanisms under various assumptions, including or not threshold trust. Many of them were covered in previous sections. But in all of them, the focus has been on existing (and well-known) mechanisms.

In another completely different line of research, first it is asked what is the relevant on-line auctions. Then a new mechanism that has all the mentioned properties is proposed. When considering the desired properties for the new auction mechanism, many variables has to be taken into account. For example, in [EL04] five properties of an ideal auction mechanism are considered:

Allocation: Usually the goal is either, *Pareto-efficiency*, which equals maximizing the social welfare (e.g. awarding the item to the bidder who valued it most); or *revenue maximization* referred to the maximization of the seller's profit.

Resource-effectiveness: The auction rules are sufficiently simple such that the seller and bidders follow them in "reasonable" time. For example, a low number of rounds is desirable.

Security against the (malicious) seller: Any behavior on benefit of seller's profit must be avoided or at least detected. As an example, the seller cannot increase the final price or change the winner without being caught.

Privacy: It refers to the information about bids and identities of bidders. A good mechanism should not allow participants to learn more information about the bids than is deducted from the result of the auction, such that winner's identity and selling price.

Minimal cognitive cost: The cognitive costs refer to the homework that bidders must do in order to give a valuation based on the information available until the moment the valuation is done.

There are many other parameters that influence the mechanism design, such as shills, collusive bids, jump biding, etc.

The cognitive costs have gained importance since other participation costs in auctions were mitigated due to the use of software agents. This was first noted by Parker, Ungar, and Foster [PUF99], where they conclude that the English auctions are the best with respect to cognitive cost when bounded-rational participants are considered. Clearly, the inclusion into the analysis of this variable in mechanism design implies to consider aspects from game theory and make assumptions on the rational behavior of the participants.

The most relevant process to 'compute' cognitive costs appears when a buyer must estimate his reservation price for a good, the maximum price that he will pay for a good, for which he doesn't know the value. The reservation price is equal to the perceived value of a good to the buyer, and depends on his preferences. There are other factors that interfere with this reservation price, but they can have no influence on the mechanism. One can trade off some of properties, and for example, the more private information is leaked, the smaller is the cognitive cost. Particularly, an English auction has low cognitive costs, while a Vickrey scheme presents a high measure for these costs.

Many authors have studied the impact of different variables in the auction mechanism design. In general, they show which mechanism is suitable in different scenarios, without giving a concrete new mechanism.

Perry *et al.*[PWZ00] constructed a two-round second price sealed-bid auction mechanism where seller's revenues are identical as English auction with the drawback that the two highest bidders of the first round obtain the distribution of the first round losers' bids. The so-called PWZ mechanism is resource-effective (with 2 rounds) and better than the Vickrey auction

in cognitive cost. However, if bidders are bounded-rational, the PWZ mechanism is not Pareto-efficient.

Proxy bidding is another designated online auction mechanism: an upper bound is fixed at first. Then bidders start an English auction until the bound has been reached. The owner of the item is consulted to continue (by setting a new bound) or not. This process can last as many rounds as the owner wants. Clearly, this mechanism has low cognitive costs, but it is not resource-effective. This mechanism presents a balance between cognitive costs and number of rounds of English and Vickrey auction mechanisms. However, proxy bidding has drawbacks from the privacy point of view: statistical information is revealed even when cryptographically secured and if participants have bounded-rational behavior, a large number of rounds is expected.

Elkind and Lipmaa [EL04] propose another interesting point of view: they first define what is relevant in auctions, and then define the mechanism. In particular, they focus on on-line auctions and conclude that security against seller and privacy is more important than cognitive costs, mainly due to the difficulty to model the latter; and cognitive costs are more relevant against computational effectiveness. Even more, the mechanism that they propose is parametrized in such a way that one can trade off the level of privacy versus cognitive costs, and cognitive costs versus computational costs. Roughly, the mechanism consists in many invocations of a Vickrey process but is not manipulable as soon as neither the Vickrey auctions are.

5.2 Fair Division

Consider a situation where a set of n players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ have to divide a set of *goods* $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$ among themselves. Every player P_i has its own, private valuation of the goods.

The goods may be *indivisible* and have to be assigned as a whole (such as a house and a car in the case of a divorce), or may be *divisible* and can be allocated in parts to the players (such as cutting a cake at a coffee table). For divisible goods, one can further distinguish the case of *homogeneous* goods from the situation where the goods are *inhomogeneous*.

Homogeneous goods are easy to divide because any fraction of a good has the same proportional value to any player, in respect to the total valuation of the good. An inhomogeneous or *heterogeneous* good, on the other hand, is harder to divide because every player may prefer to split it in a different way. With inhomogeneous goods, one usually considers only one good for simplicity, namely a *cake*, so that this area has become known as "cake cutting."

The goal of *fair division* is to split the goods in such a way such that every player receives a share of the goods that it considers to be a "fair" share of the sum. This is not trivial because every player has its individual valuation. It is not even clear which should be the appropriate measure for fairness. Fairness can be defined in the following ways:

Proportionality: Every player receives a share that it values at least $1/n$ of the total value.

Envy-freeness: Every player values its own share at least as much as the share of any other player.

Efficiency (Pareto-optimality): There is no other division that is strictly better for one player and at least as good for all others.

Equitability: The (announced) valuations of the shares of all parties are equal.

Proportionality alone is often easy to achieve, which is the reason for many works to focus on envy-freeness as their main criterion for fairness. Satisfying multiple of the advanced fairness measures simultaneously, such as envy-freeness and efficiency and equitability, is usually not possible.

There exists a rich literature on the subject, with practical applications ranging from conflict resolution in politics to court cases and contract negotiations (see [Bra05] and references therein).

It is interesting to note that one of the fundamental methods of fair division, namely the "cut and choose" procedure, has found its way into cryptographic protocols since their introduction. In the cut-and-choose method, one player divides the goods into two sets that it values equally, and the other player chooses which set to take. It creates an envy-free allocation even in the most difficult case considered here, namely, when splitting a number of heterogeneous goods.

In interactive cryptographic proof protocols, the cut-and-choose method is employed in a situation where the prover prepares two (or more) hidden statements that substantiate its claim, but if opened simultaneously would jeopardize the desired secrecy of the proof statement. The verifier of the proof then chooses one of the statements and the prover has to open it. Because the prover could cheat with probability one half, this step is then repeated sufficiently many times. The independence of the verifier's selection from the prover's statements establishes the soundness of the proof.

In practice, many if not all game-theoretic division procedures have to be computed by a trusted third party because a player knowing the valuation of its opponent can cheat the method by not using its true valuation for the procedure. This motivates the study of cryptographically secure implementations of fair division methods, which respect the privacy of the players and yet implement the division in a rational way.

5.2.1 Indivisible Goods

When the goods are indivisible, an envy-free assignment may not exist. In this case, allocations with minimal envy and approximations to envy-free allocations have been studied.

More precisely, let $v_i(S) : 2^{\mathcal{G}} \to \mathbf{R}$ be the valuation function of P_i. An allocation A is a partition of the goods $\mathcal{G}$ in sets $A_1, A_2, \ldots, A_n$, such that P_i obtains A_i. A player P_i envies a player P_k in an allocation if $v_i(A_i) < v_i(A_k)$. The envy of A_i for A_k is defined as $\max\{0, v_i(A_k) - v_i(A_i)\}$.

The marginal utility of a good G_j with respect to player P_i and a subset S of goods is defined as the amount by which P_i's valuation of S increases when G_j is added to S.

Lipton *et al.* [LMMS04] give an algorithm that allocates m goods among n players with $O(mn^3)$ time complexity and ensures that the maximum envy between every pair is at most the maximum marginal utility.

But, in general, computing or even approximating allocations with minimum envy is a hard problem. Lipton *et al.* [LMMS04] show that finding an allocation with minimum envy for all pairs of players is NP-complete. Furthermore, for any constant c, computing even a 2^{m^c} approximation to the minimum-envy allocation is impossible unless $P = NP$. On the other hand, they give an algorithm that approximates the minimum *envy ratio* between two players for the special case that all players use the same valuation.

Further improvements for allocating indivisible goods appear in [BD05, Gol05].

5.2.2 Divisible Goods

Homogeneous Divisible Goods with $n = 2$ Players

Situations that involve the division multiple homogeneous goods arise often. For *two* players, the best procedure for dividing a set of divisible homogeneous goods is the *adjusted winner* method of Brams and Taylor [BT96]. It provides envy-freeness, efficiency, and equitable divisions and therefore achieves an optimal degree of fairness for two parties, given the above notions of fairness.

The adjusted winner procedure assumes that valuations on a single good are linear and that valuations across goods are additive; linearity corresponds to our assumption that goods are homogeneous, and additivity means that the valuation of multiple goods is the sum of the individual valuations.

The method works as follows [BT96]. Given the goods $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$, both players secretly express their valuation functions, $v_1(\cdot)$ and $v_2(\cdot)$, in the form of distributing a unit valuation (say, 100 points) as bids on the goods. In a first step, every good is assigned to the player whose bid on the good is higher, i.e.,

$$\tilde{A}_1 \;=\; \big\{G_j \in \mathcal{G} | v_1(G_j) \geq v_2(G_j)\big\}$$

and

$$\tilde{A}_2 \;=\; \big\{G_j \in \mathcal{G} | v_1(G_j) < v_2(G_j)\big\}.$$

Since this usually results in an unfair allocation, an "equitability adjustment step" takes place next, in which some of P_1's allocation is transferred to P_2 until the sums of their valuations are equal. The adjustment starts with those goods that both players value similarly and continues with the goods on which their valuations differ; it stops only when both player's valuations are equal.

In other words, let π be a permutation of $\{1, \ldots, m\}$ such that

$$\frac{v_1(G_{\pi(j)})}{v_2(G_{\pi(j)})} \leq \frac{v_1(G_{\pi(j')})}{v_2(G_{\pi(j')})} \quad \text{if and only if} \quad j < j'.$$

Let $r \in \{1, \ldots, m\}$ be the index such that $\frac{v_1(G_{\pi(r)})}{v_2(G_{\pi(r)})} < 1 \leq \frac{v_1(G_{\pi(r+1)})}{v_2(G_{\pi(r+1)})}$. Hence P_1 values $G_{\pi(r+1)}, \ldots, G_{\pi(m)}$ equally or more than P_2, and P_2 values $G_{\pi(1)}, \ldots, G_{\pi(r)}$ more than P_1. Then, using good $\tilde{G} = G_{\pi(r+1)}$, $G_{\pi(r+2)}$, ..., they repeatedly transfer as much of $\tilde{G}$ from P_1 to P_2 until the valuations of P_1 and P_2 of the assignments are equal.

By design, the adjusted winner procedure is envy-free and equitable. One can also prove that it is efficient.

Because of its simple structure, the adjusted winner procedure lends itself easily to a private, cryptographic implementation among two players using Yao's protocol for secure function evaluation. It is an interesting open problem to devise a more efficient implementation.

Homogeneous Divisible Goods with $n > 2$ Players

When there are more than two players, there exist division procedures that, in general, do not yield efficient allocations. We refer to the book by Brams and Taylor for an overview [BT96].

Heterogeneous Divisible Goods — Cake Cutting

When the goods are divisible and heterogeneous, splitting them offers the most possibilities. As stated before, one models this by cutting one cake into pieces to be assigned to the n players. With more than two players, the assignment of a player no longer consists of a single piece.

A deterministic algorithm using $O(n \log n)$ cuts was devised by Even and Paz [EP84], and represents the state of the art. Computing a proportional division among n players, i.e., such that every player receives $1/n$ of the cake in its own valuation, was shown to require $\Omega(n \log N)$ cuts with any deterministic protocol by Woeginger and Sgall [WS04]. Their lower bound matches the algorithm of Even and Paz.

Using an approximation algorithm, Woeginger and Sgall [WS04] also describe a protocol that comes arbitrarily close computing a proportional division in only $O(n)$ cuts.

The formalization underlying the above results uses the unit interval $[0, 1]$ to represent the cake, for which every player has its own measure that can only be evaluated by calling an evaluation oracle supplied by the player. This model differs from the cases of indivisible and homogeneous goods considered before. Making these algorithms amenable to a secure implementation using a cryptographic protocol first requires discretization of the problem.

5.3 Collaborative Benchmarking and Forecasting

5.3.1 Introduction

Benchmarking is a process used in management in which organizations evaluate various aspects of their processes in relation to best practice. It allows organizations to develop plans on how to adopt such best practice and to improve or confirm deployed processes. Benchmarking can be executed individually or in collaboration. Collaborative benchmarking involves multiple enterprises evaluating their processes in comparison to each other, and is structured so as to enable those engaging in the process to compare their services, activities, processes, products, and results in order to identify their comparative strengths and weaknesses as a basis for self-improvement and/or regulation. It is supposed to offer a way of identifying better and smarter ways of doing things and understanding why they are better or smarter. These insights can then be used to implement changes that will improve practice or performance.

Inputs to collaborative benchmarking can include various statistical information and parameters of the underlying processes such as cash flow, time to ship, specific product information, return of investment, etc. However, these inputs usually are sensitive data that enterprises may not be willing to disclose due to various reasons, e.g., such information may be accessed by their competitors. In this context secure function evaluation (SFE) allows enterprises to perform computation over these inputs without revealing any useful information about them through the computation or the output.

A related problem is the collaborative forecasting where the involved entities can obtain a better prediction of their parameters for the future based on the statistics they input to the forecasting process. Privacy-preserving forecasting is especially advantageous for small companies since they can cooperate securely and use the outcome of the computation to make more accurate decisions in their business as larger entities about planning, production, quality control, etc.

A more general form of the problems mentioned above is establishing *secure supply chains*.

The supply chain is the linked set of resources and processes that begins with the sourcing of raw material (service) and extends through the delivery of end items to the final customer. It includes vendors, manufacturing facilities, logistics providers, internal distribution centers, distributors, wholesalers and all other entities. The extended supply chain for a given company may also include secondary vendors to their immediate vendors, and the customers of their immediate customers. *Supply chain management* (*SCM*) is the process of planning, implementing, and controlling the operations of the supply chain with the primary objective to fulfill customer demands through the most efficient use of resources.

5.3.2 Trust and Adversary Model

The main entities involved have usually different and potentially conflicting security policies and requirements. With respect to their malicious behavior they can be categorized in the following types: *Semi-honest* parties are honest-but-curious, i.e., it is assumed that they behave correctly according to the specified protocols but might try to infer security critical information in various protocol steps and phases. In contrast *malicious* parties may misbehave arbitrarily.

For both types of attackers one also considers collusion attacks where several parties collude against other players attempting to violate the security requirements of those parties.

Depending on the application one may or may not consider malicious adversaries. Basically, one may assume that the involved parties are interested in the correct outcome of the computation and would not deliberately manipulate the computation. Hence only semi-honest parties and their collusions need to be considered when designing protocols for benchmarking and forecasting. However, collusion attacks should still be considered since the underlying trust model assumes mistrusting entities with "commercial sensitivity". This is the approach made in [ABL+04].

Nevertheless, one can also think of situations where a malicious adversary deviates from the protocol specification with the goal to infer useful information about the sensitive data of the other entities without having any impact on the final outcome. This type of attacks usually differ from honest-but-curious attempts.

5.3.3 Related Work

Collaborative forecasting and benchmarking has already been identified as very helpful method and has been studied by enterprises, organizations and academia (see, e.g., [Ste93, KLB97, STM, Sin02, SFM04]). Most solutions either assume existence of a trusted central party with access to all information, or assume sharing of information with other involved entities. These solutions are often difficult when potential benchmarking partners are also competitors, as "commercial sensitivity" often prevents them from revealing details of their processes.

As mentioned above, secure collaborative forecasting and benchmarking is related to SFE. Protocols for secure benchmarking and forecasting in semi-honest adversary model have been proposed in [ABL+04] which we explain briefly in the following where we use the same notation from [ABL+04] for brevity. The forecasting methods that are used as the basis for protocols construction are *time-series techniques* and *regression techniques* [Ste93, Eva93]: In the time-series techniques, a time series is a time-ordered sequence of observations taken at regular intervals over a period of time (daily, weekly, monthly, annually, etc.). Let i denote the i-th

time period, and d_i denote data in time period i. Further, let t be the current time period. Then [ABL+04] considers the following three methods:

- *Moving Average*: Let n denote the number of periods used in calculation of the average. For time period $t \geq n$, the moving average forecast is: $F_t = (\sum_{i=0}^{n-1} d_{t-i})/n$ where F_t indicates the forecasted value for time interval $t+1$.
- *Weighted Moving Average*: Let $W := (w_0, w_1, \cdots, w_{n-1})$ be a weight vector such that $\sum_{i=0}^{n-1} w_i = 1$. For time period $t \geq n$, the weighted moving average forecast is $F_t = \sum_{i=0}^{n-1} w_i d_{t-i}$
- *Exponential Smoothing*: Let F_i be the forecasted value in time period i, and α be a smoothing constant. For time period t, the exponential smoothing technique is defined as $F_t := F_{t-1} + \alpha(d_{t-1} - F_{t-1})$ where F_t is also the predicted value for the next time period.

The regression method taken in [ABL+04] is the widely used *linear regression* were one assumes that two variables with linear correlation are given and the goal is to compute a linear function such that the sum of the deviations of all the points from the function is minimized. Let a linear function $y = ax + b$ where all data points x are known. If there are historical data under the form of n pairs (x, y), then after applying regression to them, one is able to estimate the coefficients a and b as follows: $a = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$ and $b = \frac{\sum y - a(\sum x)}{n}$.

In [KT06] the authors propose a filter for benchmarking that runs before the actual protocol that computes the statistical quantity. The goal is to restrict the computation to the k best values. The main reason is to avoid distortion of the outcome through weak inputs.

Other business models that have been considered in this context include scientific computation [DA01a], geometric computation [AD01], and statistics analysis [DA01b].

Another related topic is *secure supply chain*. In [AEDS03] the authors consider the *secure supply chain collaboration* (*SSCC*) problem, and propose SSCC protocols for simple e-Auction scenarios and simple capacity-allocation problem.

As mentioned before, the primary goal of supply chain management is to make an efficient use of the resources in a supply chain or in general to optimize a certain function. Whereas original supply chain management is performed in a centralized manner, today, different parties involved have own policies and incentives regarding security and efficiency/profit. Here a new situation arises since the same information may not be available to all parties, i.e., there is an information *asymmetry*, and the involved parties do not want to share this information with each other even if this would be needed for the desired system-wide goals. Hence compared to the traditional scenarios each party P_i has its own private function $f_{P_i}()$, and the decision this party makes is based on the outcome of the function which depends not only on the private inputs x_{P_j} of other parties but also on the private function f_{P_i}.

5.4 Polling

The goal of polling is to estimate a random variable through sampling: a set of persons are selected in a large group, and these persons are asked to answer the same question; the answers are then used to estimate the opinion of the whole group. The goals of a poll differ from those of a vote by at least two important aspects: first, one only needs to estimate a

random variable, while an exact count of the responses is required from a voting protocol. Second, even if some degree of privacy might be useful to make the responders feel more comfortable in answering honestly, the possibility to deny an answer is usually a sufficient guarantee, while strong secrecy properties need to be guaranteed in voting protocols.

In order to gain some privacy, Warner [War65] proposed to use a "randomized response technique". The idea is to tell responders to lie with some fixed, predetermined, probability, which gives the responders an easy way to deny their response, while retaining the ability of the pollster to evaluate the polled variable.

For instance, suppose $p > \frac{1}{2}$ is the known probability of a truthful response, n is the total number of responses, x is the number of responders who actually belong in the "yes" category and R is the random variable counting the number of "yes" responses. Then, R is a good estimation for $E(R) = px + (1-p)(n-x)$. Therefore, we can estimate x as $\frac{E(R)-n(1-p)}{2p-1}$.

The randomized response technique has been declined in many flavors, offering different compromises between usability and precision of the results (a review of these techniques is available in [CM88]).

Randomized response may raise some problems when players have an incentive to influence the poll, which may happen in polls preceding elections for instance, where one can expect that the result of the poll will influence the outcome of the elections (various types of influences have been identified: the "bandwagon effect", where undecided persons tend to actually vote for the candidate leading the polls, the "boomerang effect" where people confident in the result of an election decide to stay home and to not vote, resulting in another candidate to win, ...). In these cases, players may decide to ignore the "lying" recommendation and always provide the answer they want to promote, which will introduce a bias in the estimation. In this context, playing the polling protocol honestly is in opposition with the rational behavior of the players.

Several works deal with this malicious bias issue [KAGN99, AJL04, MN06]. The idea here is to give the pollster a way to force the responder to use some fixed randomness, as expected in the protocol definition. As in the rational protocol of Izmalkov et al. [IML05], the only way a malicious responder can cheat becomes to abort the protocol when he sees that his vote will not be interpreted in the way he wants to promote. However, this can be easily observed by the pollster.

Kikuchi *et al.* [KAGN99] force randomization by using a fair coin-tossing protocol. Ambainis *et al.* [AJL04] improve on this result, providing formal definitions for cryptographic randomized response, and obtaining a more efficient (and also more complicate) solution based on an oblivious transfer protocol and noninteractive ZK arguments. Recently, Moran and Naor proposed elegant solutions based on envelopes with tamper-evident seals [MN06], and proved the security of their protocols in the universally composable security framework [Can01]. The simplicity of the protocols of Moran and Naor is very appealing: their protocols can be executed by human, without requiring the help of computers and, in many cases, involve very few interactions. It would be very interesting to investigate how they can be further improved, in terms of flexibility, reliability, and number of required interactions.

Bibliography

[ABL+04] M. J. Atallah, M. V. Bykova, J. Li, K. B. Frikken, and M. Topkara, *Private collaborative forecasting and benchmarking*, 3rd annual Workshop on Privacy in the Electronic Society (WPES 04), 2004.

[AD01] M. Atallah and W. Du, *Secure multi-party computational geometry*, International Workshop on Algorithms and Data Structures (WADS2001), 2001, pp. 165–179.

[ADGH06] I. Abraham, D. Dolev, R. Gonen, and J. Halpern, *Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation*, PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, ACM Press, 2006, pp. 53–62.

[AEDS03] M. Atallah, H. Elmongui, V. Deshpande, and L. Schwarz, *Secure supply-chain protocols*, IEEE International Conference on Electronic Commerce, 2003, pp. 293–302.

[AFG+05] G. Aggarwal, A. Fiat, A. V. Goldberg, J. D. Hartline, N. Immorlica, and M. Sudan, *Derandomization of auctions*, Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005, pp. 619–625.

[AJL04] A. Ambainis, M. Jakobsson, and H. Lipmaa, *Cryptographic randomized response techniques*, DIMACS/PORTIA Workshop on Privacy-Preserving Data Mining, 2004.

[AS02a] M. Abe and K. Suzuki, *$(m+1)$-st price auction using homomorphic encryption*, Public Key Cryptography—PKC 2002, Lecture Notes in Computer Science, vol. 2274, Springer-Verlag, 2002, pp. 115–224.

[AS02b] ———, *Receipt-free sealed-bid auction*, 1st Information Security Conference—ISC 2002, Lecture Notes in Computer Science, vol. 2433, Springer-Verlag, 2002, pp. 191–199.

[Aum74] R. Aumann, *Subjectivity and correlation in randomized strategies*, J. Math. Econ. **1** (1974), 67–96.

[Bar86] D. A. M. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC*, Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC), 1986, pp. 1–5.

[Bár92] I. Bárány, *Fair distribution protocols or how the players replace fortune*, Mathematics of Operation Research **17** (1992), 327–341.

[BD05] I. Bezáková and V. Dani, *Allocating indivisible goods*, ACM SIGecom Exchanges **5** (2005), no. 3, 11–18.

[BDJ+06] P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft, *A practical implementation of secure auctions based on multiparty integer computation.*, Financial Cryptography (G. D. Crescenzo and A. Rubin, eds.), Lecture Notes in Computer Science, vol. 4107, Springer, 2006, pp. 142–147.

[Ben98] E. Ben-Porath, *Correlation without mediation: Expanding the set of equilibrium outcomes by "cheap" pre-play procedures*, Journal of Economic Theory **80** (1998), no. 1, 108–122.

[Bra02] F. Brandt, *A verifiable, bidder-resolved auction protocol*, 5th AAMAS Workshop on Deception, Fraud and Trust in Agent Societies (Special Track on Privacy and Protection with Multi-Agent Systems), 2002, pp. 18–25.

[Bra03a] ———, *Fully private auctions in a constant number of rounds*, Financial Cryptography 2003, Lecture Notes in Computer Science, vol. 2742, Springer-Verlag, 2003, pp. 223–238.

[Bra03b] ———, *Fundamental aspects of privacy and deception in electronic auctions*, Ph.D. thesis, Technical University of Munich, 2003.

[Bra05] S. J. Brams, *Fair division*, Oxford Handbook of Political Economy (B. R. Weingast and D. Wittman, eds.), Oxford University Press, 2005, Preliminary version at `http://www.nyu.edu/gsas/dept/politics/faculty/brams/fd_handbook.pdf`.

[BS04] H. K. Bhargava and S. Sundaresan, *Contingent bids in auctions: Availability, commitment and pricing of computing as utility*, Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 8, IEEE Computer Society, 2004, p. 80217.3.

[BT96] S. J. Brams and A. D. Taylor, *Fair division*, Cambridge University Press, 1996.

[BW01] F. Brandt and G. Weiß, *Antisocial agents and vickrey auctions*, Pre-proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), 2001, pp. 120–132.

[Cac99] C. Cachin, *Efficient private bidding and auctions with an oblivious third party*, 6th ACM conference on Computer and Communications Security–CCS '99, ACM Press, 1999, pp. 120–127.

[Can00] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols*, Cryptology ePrint Archive 2000/067, 2000.

[Can01] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols*, 42nd Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 2001, Full version in [Can00], pp. 136–145.

[CBK03] X. Chen, B., and K. Kim, *Receipt-free electronic auction schemes using homomorphic encryption*, International Conference Information Security and Cryptology—ICISC2003, Lecture Notes in Computer Science, vol. 2971, 2003, pp. 259–273.

[CD05] X. Chen and X. Deng, *Settling the complexity of 2-player nash-equilibrium*, Electronic Colloquium on Computational Complexity (ECCC), Report TR05-140, 2005.

[CDN01] R. Cramer, I. Damgård, and J. Nielsen, *Multiparty computation from threshold homomorphic encryption*, Advances in Cryptology—Eurocrypt 2001, Lecture Notes in Computer Science, vol. 2045, Springer-Verlag, 2001, pp. 280–300.

[Cle86] R. Cleve, *Limits on the security of coin flips when half the processors are faulty (extended abstract)*, Proceedings of the 18th ACM Symposium on Theory of Computing (STOC), 1986, pp. 364–369.

[CM88] A. Chaudhuri and R. Mukerjee, *Randomized response: Theory and techniques*, Statistics: Textbooks and Monographs, vol. 95, Marcel Dekker, Inc., 1988.

[CSS06] P. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial auction*, MIT Press, 2006.

[DA01a] W. Du and M. Atallah, *Privacy-preserving cooperative scientific computations*, IEEE Computer Security Foundations Workshop, 2001, pp. 273–282.

[DA01b] ———, *Privacy-preserving statistical analysis*, Annual Computer Security Applications Conference, 2001, pp. 102–110.

[DFK+06] I. Damgård, M. Fitzi, E. Kiltz, J. Nielsen, and T. Toft, *Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation*, Theory of Cryptography Conference—TCC 2006, Lecture Notes in Computer Science, vol. 3876, Springer-Verlag, 2006, pp. 285–304.

[DGP06] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, *The complexity of computing a Nash equilibrium*, Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), 2006, pp. 71–78.

[DHR00] Y. Dodis, S. Halevi, and T. Rabin, *A cryptographic solution to a game theoretic problem.*, Advances in Cryptology—Crypto 2000 (M. Bellare, ed.), vol. 1880, Springer-Verlag, 2000, Lecture Notes in Computer Science, pp. 112–130.

[DoJ00] *U.S. Department of Justice Primer. Price Fixing, Bid Rigging, and Market Allocation Schemes: What They Are and What to Look For*, 2000, `http://www.usdoj.gov/atr/public/guidelines/pfbrprimer.pdf`.

[EL04] E. Elkind and H. Lipmaa, *Interleaving cryptography and mechanism design: The case of online auctions.*, Financial Cryptography (A. Juels, ed.), Lecture Notes in Computer Science, vol. 3110, Springer, 2004, pp. 117–131.

[EP84] S. Even and A. Paz, *A note on cake cutting*, Discrete Applied Mathematics **7** (1984), 285–296.

[Eva93] J. Evans, *Applied production and operations management*, 4th edition ed., West Publishing Company, 1993.

[FGHK02] A. Fiat, A. V. Goldberg, J. D. Hartline, and A. R. Karlin, *Competitive generalized auctions*, STOC, ACM, 2002, pp. 72–81.

[FLP85] M. J. Fischer, N. A. Lynch, and M. S. Paterson, *Impossibility of distributed consensus with one faulty process*, Journal of the ACM **32** (1985), no. 2, 374–382.

[FS90] U. Feige and A. Shamir, *Witness indistinguishable and witness hiding protocols*, STOC, ACM, 1990, pp. 416–426.

[GHKS04] A. V. Goldberg, J. D. Hartline, A. R. Karlin, and M. Saks, *A lower bound on the competitive ratio of truthful auctions*, STACS, Lecture Notes in Computer Science, vol. 2996, Springer, 2004, pp. 644–655.

[GK06] S. D. Gordon and J. Katz, *Rational secret sharing revisited*, Security in Communication Networks (SCN) (R. D. Prisco and M. Yung, eds.), LNCS, Springer-Verlag, 2006.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, SIAM Journal on Computing **18** (1989), no. 1, 186–208.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game or a completeness theorem for protocols with honest majority*, 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, pp. 218–229.

[Gol04] O. Goldreich, *Foundations of cryptography*, vol. I & II, Cambridge University Press, 2001–2004.

[Gol05] D. Golovin, *Max-min fair allocation of indivisible goods*, Technical Report CMU-CS-05-144, School of Computer Science, Carnegie-Mellon University, 2005.

[GP06] P. W. Goldberg and C. H. Papadimitriou, *Reducibility among equilibrium problems*, Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), 2006, pp. 61–70.

[GSV07] J. Garay, B. Schoenmakers, and J. Villegas, *Practical and secure solutions for integer comparison*, Public Key Cryptography—PKC 2007, Lecture Notes in Computer Science, 2007, To appear.

[Han85] R. Hansen, *Auctions with contingent payments*, American Economic Review **75** (1985), 862–65.

[HIS05] Y. Her, K. Imamoto, and K. Sakurai, *Some remarks on security of receipt-free e-auction*, 3rd International Conference on Information Technology and Applications—ICITA'05, vol. 2, IEEE Computer Society, 2005, pp. 560–563.

[HT04] J. Y. Halpern and V. Teague, *Rational secret sharing and multiparty computation: extended abstract*, Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), 2004, pp. 623–632.

[IML05] S. Izmalkov, S. Micali, and M. Lepinski, *Rational secure computation and ideal mechanism design*, 46th IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 585–595.

[JJ00] M. Jakobsson and A. Juels, *Mix and match: Secure function evaluation via ciphertexts*, Advances in Cryptology—Asiacrypt 2000, Lecture Notes in Computer Science, vol. 1976, Springer-Verlag, 2000, pp. 162–177.

[Jur00] M. Jurdziński, *Games for verification: Algorithmic issues*, Dissertation series, University of Aarhus, Department of Computer Science, University of Aarhus, December 2000, PhD thesis.

[KAGN99] H. Kikuchi, J. Akiyama, H. Gobioff, and G. Nakamura, *Stochastic voting protocol to protect voters privacy*, IEEE Workshop on Internet Applications, 1999, pp. 103–111.

[KLB97] A. Khetawat, H. Lavana, and F. Brglez, *Collaborative workflows: A paradigm for distributed benchmarking and design on the internet*, Tech. Report 97-TR@CBL-02, CS Dept., North Carolina State University, Raleigh, 1997.

[KP99] E. Koutsoupias and C. H. Papadimitriou, *Worst-case equilibria*, STACS (C. Meinel and S. Tison, eds.), Lecture Notes in Computer Science, vol. 1563, Springer, 1999, pp. 404–413.

[KS03] J. Kim and A. Segev, *Multi-component contingent auction (MCCA): A procurement mechanism for dynamic formation of supply networks*, 5th International Conference on Electronic Commerce–ICEC '03, 2003, pp. 78–86.

[KT06] F. Kerschbaum and O. Terzidis, *Filtering for private collaborative benchmarking*, International Conference on Emergin Trends in Information and Communication Security, LNCS 3995, 2006.

[KW82] D. Kreps and R. Wilson, *Sequential equilibria*, Econometrica **50** (1982), 863–894.

[LAN02] H. Lipmaa, N. Asokan, and V. Niemi, *Secure vickrey auctions without threshold trust*, Financial Cryptography 2002, Lecture Notes in Computer Science, vol. 2357, Springer-Verlag, 2002, pp. 87–101.

[LMMS04] R. Lipton, E. Markakis, E. Mossel, and A. Saberi, *On approximately fair allocations of indivisible goods*, 5th ACM Conference on Electronic Commerce (EC), 2004.

[LMPS04] M. Lepinski, S. Micali, C. Peikert, and A. Shelat, *Completely fair SFE and coalition-safe cheap talk*, PODC (S. Chaudhuri and S. Kutten, eds.), ACM, 2004, pp. 1–10.

[LMS04] M. Lepinski, S. Micali, and A. Shelat, *Fair zero-knowledge*, Theory of Cryptology Conference, LNCS, Springer, 2004.

[LMS05] ———, *Collusion-free protocols*, Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005, pp. 543–552.

[LP04] Y. Lindell and B. Pinkas, *A proof of Yao's protocol for secure two-party computation*, Electronic Colloquium on Computational Complexity (ECCC), Report TR04-063, 2004.

[LT87] N. A. Lynch and M. R. Tuttle, *Hierarchical correctness proofs for distributed algorithms*, Tech. report, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1987, MIT/LCS/TR-387.

[LT06] A. Lysyanskaya and N. Triandopoulos, *Rationality and adversarial behavior in multi-party computation*, Advances in Cryptology—Crypto 2006 (C. Dwork, ed.), vol. 4117, Springer-Verlag, 2006, Lecture Notes in Computer Science, pp. 180–197.

[MCW05] C. Meek, D. M. Chickering, and D. B. Wilson, *Stochastic and contingent payment auctions*, Workshop on Sponsored Search Auctions, ACM Electronic Commerce, 2005.

[MN05] T. Moran and M. Naor, *Basing cryptographic protocols on tamper-evident seals.*, Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings (L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, eds.), Lecture Notes in Computer Science, vol. 3580, Springer, 2005, pp. 285–297.

[MN06] ———, *Polling with physical envelopes: A rigorous analysis of a human-centric protocol*, Eurocrypt, 2006, pp. 88–108.

[MS06] P. B. Miltersen and T. B. Srensen, *Computing sequential equilibria for two-player games*, SODA, ACM Press, 2006, pp. 107–116.

[Nas51] J. Nash, *Non-cooperative games*, Annals of Mathematics **54** (1951), 286–295.

[NPS99] M. Naor, B. Pinkas, and R. Sumner, *Privacy preserving auctions and mechanism design*, 1st ACM Conference on Electronic Commerce, 1999.

[NR99] N. Nisan and A. Ronen, *Algorithmic mechanism design*, 31st ACM Symposium on Theory of Computing (STOC), 1999, pp. 129–140.

[Omo02] K. Omote, *A study on electronic auctions*, 2002.

[Pap05] C. H. Papadimitriou, *Computing correlated equilibria in multi-player games*, Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005.

[PR05] C. H. Papadimitriou and T. Roughgarden, *Computing equilibria in multi-player games*, Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, ACM Press, 2005, pp. 82–91.

[PSL80] M. Pease, R. Shostak, and L. Lamport, *Reaching agreement in the presence of faults*, Journal of the ACM **27** (1980), no. 2, 228–234.

[PUF99] D. Parkes, L. Ungar, and D. Foster, *Accounting for cognitive costs in on-line auction design*, Agent Mediated Electronic Commerce. First International Workshop on Agent Mediated Training, AMET '98, vol. 1571, Springer-Verlag, 1999, pp. 25–40.

[PWZ00] M. Perry, E. Wolfstetter, and S. Zamir, *A sealed-bid auction that matches the english auction*, Games and Economic Behaviour **33** (2000), 265–273.

[RKV00] M. Rhodes-Kropf and S. Viswanathan, *Corporate reorganizations and non-cash auctions*, The Journal of Finance **55** (2000), 1807–1854.

[Rou03] T. Roughgarden, *The price of anarchy is independent of the network topology*, J. Comput. Syst. Sci **67** (2003), no. 2, 341–364.

[Rou05] ———, *Selfish routing and the price of anarchy*, MIT Press, 2005.

[RT00] T. Roughgarden and É. Tardos, *How bad is selfish routing?*, 41st IEEE Symposium on Foundations of Computer Science (FOCS), 2000, pp. 93–102.

[Sel65] R. Selten, *Spieltheoretische Behandlung eines Oligopolmodells mit Nachfrageträgheit*, Zeitschrift für die gesamte Staatswissenschaft **12** (1965), 301–324.

[SFM04] L. Smith, J. Flannery, and G. Maxwell, *Collaborative planning, forecasting, and replenishment (CPFR)*, available at `http://www.cpfr.org/Members.html`, 2004.

[Sha79] A. Shamir, *How to share a secret*, Communications of the ACM **22** (1979), no. 11, 612–613.

[Sin02] H. Singh, *Collaborative forecasting*, available at `http://www.supplychain.com/docs/collaborativeforecasting.pdf`, 2002.

[ST04] B. Schoenmakers and P. Tuyls, *Practical two-party computation based on the conditional gate*, Advances in Cryptology—Asiacrypt 2004, Lecture Notes in Computer Science, vol. 3329, Springer-Verlag, 2004, pp. 119–136.

[ST05] Y. Shoham and M. Tennenholtz, *Non-cooperative computation: Boolean functions with correctness and exclusivity*, Theor. Comput. Sci. **343** (2005), no. 1-2, 97–113.

[Ste93] W. Stevenson, *Production/operations management*, 4th edition ed., Richard D. Irwin, Inc., 1993.

[STM] STM, *Achieving streamlined operations through collaborative forecasting and inventory management*, available at `https://portal.rosettanet.org/cms/export/sites/default/RosettaNet/Downloads/ROIStudies/STMicroelectronics_White_Paper.pdf`.

[Tea04] V. Teague, *Selecting correlated random actions*, Financial Cryptography (A. Juels, ed.), Lecture Notes in Computer Science, vol. 3110, Springer, 2004, pp. 181–195.

[UPF98] L. Ungar, D. Parkes, and D. Foster, *Cost and trust issues in on-line auctions*, Workshop on Agent-Mediated Electronic Trading (AMET '98), 1998, pp. 161–172.

[Vic61] W. Vickrey, *Counterspeculation, auctions, and competitive sealed tenders*, Journal of Finance **16** (1961), 8–37.

[War65] S. Warner, *Randomized response: a survey technique for eliminating evasive answer bias*, Journal of the American Statistical Association (1965), 63–69.

[WS04] G. J. Woeginger and J. Sgall, *On the complexity of cake cutting*, Manuscript, available from `http://www.math.cas.cz/~sgall/ps/cake.ps`, 2004.

[Yao82] A. C. C. Yao, *Theory and application of trapdoor functions*, IEEE Symposium on Foundations of Computer Science, 1982, pp. 80–91.

[Yao86] ———, *How to generate and exchange secrets*, 27th IEEE Symposium on the Foundations of Computer Science (FOCS), 1986, pp. 162–167.

Index