Efficient Rational Secret Sharing in Standard Communication Networks

Georg Fuchsbauer* Jonathan Katz† Eric Levieil* David Naccache*

Abstract

We propose a new methodology for rational secret sharing leading to various instantiations that are simple and efficient in terms of computation, share size, and round complexity. Our protocols do not require physical assumptions or simultaneous channels, and can even be run over asynchronous, point-to-point networks.

Of additional interest, we propose new equilibrium notions for this setting (namely, computational versions of *strict Nash equilibrium* and *stability with respect to trembles*) and prove that our protocols satisfy them.

1 Introduction

The classical problem of *t-out-of-n secret sharing* [27, 5] involves a dealer D who distributes shares of a secret s to a group of n players $P_1, \ldots, P_n$ so that (1) any group of t or more players can reconstruct the secret without further involvement of the dealer, yet (2) any group of fewer than t players cannot recover the secret. For example, in Shamir's scheme [27] the secret s lies in a finite field $\mathbb{F}$, with $|\mathbb{F}| > n$. The dealer chooses a random polynomial $f(x)$ of degree at most $t-1$ with $f(0) = s$, and gives each player P_i the "share" $f(i)$. To reconstruct the secret s, any t players simply broadcast their shares and interpolate the polynomial. On the other hand, any set of fewer than t players has no information about s given their shares.

The implicit assumption in the original formulation of the problem is that each party is either honest or corrupt, and honest parties are all willing to cooperate when reconstruction of the secret is desired. Beginning with the work of Halpern and Teague [12], protocols for secret sharing and other cryptographic tasks have begun to be re-evaluated in a game-theoretic light (see [7, 15] for an overview of work in this direction). In this setting, parties are neither honest nor corrupt but are instead viewed as *rational* and are assumed (only) to act in their own self-interest.

Under natural assumptions regarding the utilities of the parties, standard secret-sharing schemes completely fail. For example, assume as in [12] that all players want to learn the secret above all else, but otherwise prefer that no other players learn the secret. (Later, we will treat the utilities of the players more precisely.) For t parties to reconstruct the secret in Shamir's scheme, each party

*École Normale Supérieure, LIENS - CNRS - INRIA, Paris, France. Email: {georg.fuchsbauer, eric.levieil, david.naccache}@ens.fr. The first author is supported by EADS, the French ANR-07-SESU-008-01 PAMPA Project, and the European Commission through the IST Program under Contract ICT-2007-216646 ECRYPT II.

†University of Maryland, USA. Email: jkatz@cs.umd.edu. Work done while visiting ENS and IBM, and supported by NSF CyberTrust grant #0830464, NSF CAREER award #0447075, and DARPA. The contents of this paper do not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

is supposed to broadcast their share simultaneously. It is easy to see, however, that each player is better off withholding their share no matter what the other players do. Consider P_1: If fewer than $t-1$ other players reveal their shares, then P_1 does not learn the secret regardless of whether P_1 reveals his share or not. If more than $t-1$ other players reveal their shares, then everyone learns the secret and P_1's actions again have no effect. On the other hand, if *exactly* $t-1$ other players reveal their shares, then P_1 learns the secret (using his share) but prevents other players from learning the secret by *not* publicly revealing his own share. The result is that if all players are rational then no one will broadcast their share and the secret will not be reconstructed.

A series of recent works [12, 10, 23, 1, 17, 18, 26, 25, 4] has focused on designing *rational* secret-sharing protocols immune to the above problem. Protocols for rational secret sharing also follow from the more general results of Lepinski et al. [19, 20, 14, 13]. Each of these works has some or all of the following disadvantages:

On-line dealer or trusted/honest parties. Halpern and Teague [12] introduced a general approach to solving the problem that has been followed in most subsequent work. Their solution, however, requires the continual involvement of the dealer, even after the initial shares have been distributed. The solution proposed by Halpern and Teague also applies only when $t, n \geq 3$.

Recent work of [13, 25] requires the involvement of some (minimally trusted) external parties during the reconstruction phase. Ong et al. [26] assume that sufficiently many parties behave honestly during the reconstruction phase.

Computational inefficiency. To eliminate the on-line dealer, several researchers [10, 23, 1, 17] have suggested solutions that rely on multiple invocations of protocols for generic secure multi-party computation. Because the function being computed by these protocols is complex, it is unclear whether computationally *efficient* protocols with suitable functionality can be designed. The solutions of [19, 20, 14, 13], though following a different high-level approach, also rely on generic secure multi-party computation.

Non-standard communication models. The solutions in [12, 10, 23, 1] assume *simultaneous broadcast* which means that parties must decide on what value (if any) to broadcast in a given round before observing the values broadcast by other parties. The solutions of [19, 20, 14] rely on *physical* assumptions such as secure envelopes and ballot boxes. Secure envelopes imply simultaneous broadcast (but not vice versa) and hence represent a strictly stronger class of assumptions.

Kol and Naor [17] show how to avoid simultaneous broadcast, at the cost of increasing the round complexity by a (multiplicative) factor linear in the size of the domain from which the secret is chosen; their approach thus has super-polynomial complexity for secrets of super-logarithmic length. Subsequent work by Kol and Naor [18] (see also [4]) shows how to avoid the assumption of simultaneous broadcast at the expense of increasing the round complexity by a (multiplicative) factor of t. We provide a detailed comparison of our results to those of [18] in Section 1.2.1.

As far as we are aware, all prior schemes for $n > 2$ assume the existence of broadcast (whether simultaneous or not).

1.1 Our Results

Our solutions do not suffer from any of the drawbacks mentioned above. We do not assume an on-line dealer or any trusted/honest parties, nor do we resort to generic secure multi-party computation. Our protocols are (arguably) simpler than previous solutions; they are also extremely efficient in terms of round complexity, share size, and required computation. Although our protocols do not require simultaneous channels (as discussed below), the efficiency advantages of our protocols hold even in comparison to prior work that *does* require simultaneous channels.

As an added benefit, our protocols also do not require simultaneous communication but can instead rely on synchronous (but *non*-simultaneous) point-to-point channels. To the best of our knowledge, all prior schemes for $n > 2$ assume broadcast (whether simultaneous or not); note that the obvious approach of simulating broadcast by running a broadcast protocol over a point-to-point network will not, in general, work in the rational setting. Moreover, we show that our protocol can be adapted to work even in *asynchronous* point-to-point networks. We thus answer a question that had been open since the work of Halpern and Teague [12].

As an independent contribution, we also introduce two new equilibrium notions and prove that our protocols satisfy them. (A discussion of game-theoretic equilibrium notions used in this and prior work is given in Section 2.2. We stress that our protocol also satisfies the notion of "surviving iterated deletion" considered in other work [12, 10].) The first notion we introduce is a *computational* version of strict Nash equilibrium. A similar notion was propounded by Kol and Naor [18], but they used an *information-theoretic* version of strict Nash and showed some inherent limitations of doing so. As in all of cryptography, we believe computational relaxations are meaningful and should be considered; this also allows us to circumvent the limitations that hold in the information-theoretic case.

Motivated by [15], we also formalize a notion of *stability with respect to trembles*; a different formalization of this notion, with somewhat different motivation, is given in [26].

An interesting feature of our definitions is that they effectively rule out "signalling" via subliminal channels in the protocol. In fact, at every point in our protocols *there is a* ***unique*** *legal message each party can send*. This prevents a party from outwardly appearing to follow the protocol while subliminally communicating (or trying to organize collusion) with other parties. Preventing subliminal communication is an explicit goal of some prior work (e.g., [14, 20, 3, 2]), which achieved it only by relying on non-standard communication models.

1.2 Overview of Our Approach

We follow the same high-level approach as in [12, 10, 23, 1, 17, 18, 4]. Our reconstruction protocol proceeds in a sequence of "fake" iterations followed by a single "real" iteration. Roughly speaking, these satisfy the following requirements:

- In the real iteration, everyone learns the secret (assuming everyone follows the protocol).
- In a fake iteration, no information about the secret is revealed.
- No party can tell, in advance, whether the next iteration will be real or fake.

The iteration number i^* of the real iteration is chosen according to a geometric distribution with parameter $\beta \in (0, 1)$ (where β depends on the players' utilities). To reconstruct the secret, parties run a sequence of iterations until the real iteration is identified, at which point all parties output the secret. If some party fails to follow the protocol, all parties abort. Intuitively, it is rational for P_i to follow the protocol as long as the expected gain of deviating, which is positive only if P_i aborts *exactly* in iteration i^*, is outweighed by the expected loss if P_i aborts before iteration i^*.

In most prior work [10, 23, 1, 17], a secure multi-party computation was performed in each iteration to determine whether the given iteration should be real or fake. Instead we use the following approach, described in the 2-out-of-2 case (we omit some technical details in order to focus on the main idea): The dealer D chooses i^* from the appropriate distribution *in advance*, at the time of sharing. The dealer then generates two key-pairs (vk_1, sk_1), (vk_2, sk_2) for a *verifiable random function* (VRF) [24], where vk represents a verification key and sk represents a secret key, and we denote by $\mathsf{VRF}_{sk}(x)$ the evaluation of the VRF on input x using secret key sk. (See Section 2.4 for formal definitions.) The dealer gives the verification keys to both parties, gives sk_1 to P_1, and gives sk_2 to P_2. It also gives $s_1 = s \oplus \mathsf{VRF}_{sk_2}(i^*)$ to P_1, and $s_2 = s \oplus \mathsf{VRF}_{sk_1}(i^*)$ to P_2. Each iteration consists of one message from each party: in iteration i, party P_1 sends $\mathsf{VRF}_{sk_1}(i)$ while P_2 sends $\mathsf{VRF}_{sk_2}(i)$. Observe that a fake iteration reveals nothing about the secret, in a computational sense. Furthermore, neither party can identify the real iteration in advance. (The description above relies on VRFs. We show that, in fact, trapdoor permutations suffice.)

To complete the protocol, we need to provide a way for parties to identify the real iteration. Previous work allows parties to identify the real iteration *as soon as it occurs*. We could use this approach for our protocol as well if we were content to assume simultaneous channel, since then each party must decide on its current-iteration message before it learns whether the current iteration is real or fake. When simultaneous channels are not available, however, this approach is vulnerable to an obvious rushing strategy. Kol and Naor [17, 18] show two different ways to avoid simultaneous broadcast, but the first applies only for secrets from polynomial-size domains (and yields round complexity linear in the domain size), while the second yields round complexity linear in t.

Motivated by recent work on fairness (in the malicious setting) [9, 11], we suggest the following, new approach: delay the signal indicating whether a given iteration is real or fake until the *following* iteration. As before, a party cannot risk aborting until it is sure that the real iteration has occurred; the difference is that now, once a party learns that the real iteration occurred, the real iteration is over and all parties can reconstruct the secret. This eliminates the need for simultaneous channels, while adding only a single round. This approach can be adapted for t-out-of-n secret sharing and can be shown to work even when parties communicate over asynchronous, point-to-point channels.

A drawback of our protocol is that it assumes parties have no auxiliary information about the secret s. Prior work in the non-simultaneous model [17, 18] shares this disadvantage, and in fact the results of [4] can be extended to show that this is inherent. (If simultaneous channels are assumed, then our protocol *does* tolerate auxiliary information about s as in previous work.)

1.2.1 Comparison to the Kol-Naor Scheme

The only prior rational secret-sharing scheme that assumes no honest parties, is computationally efficient, and does not require simultaneous broadcast or physical assumptions is that of Kol and Naor [18]. They also use the strict Nash solution concept and so their work provides an especially good point of comparison. Our protocols have the following advantages with respect to theirs:

Share size. In the Kol-Naor scheme, the shares of the parties have *unbounded* length. While not a significant problem in its own right, this is problematic when rational secret sharing is used as a sub-routine for rational computation of general functions. (See [17].) Moreover, the *expected* length of the parties' shares in their scheme is large: in the 2-out-of-2 case, shares of a secret s have expected size $O(\beta^{-1} \cdot (|s| + k))$ in the Kol-Naor scheme (where k is a security parameter), whereas shares in our scheme have size $|s| + O(k)$.

Round complexity. The version of the Kol-Naor scheme that does not rely on simultaneous broadcast [18, Section 6] has expected round complexity $O(\beta^{-1} \cdot t)$, whereas our protocol has expected round complexity $O(\beta^{-1})$. (The value of β is roughly the same in both cases.)

Resistance to coalitions. For the case of t-out-of-n secret sharing, the Kol-Naor scheme is susceptible to coalitions of two or more players. We show t-out-of-n secret-sharing protocols resilient to coalitions of up to $(t-1)$ parties; see Section 4 for further details.

Avoiding broadcast. The Kol-Naor scheme for $n > 2$ assumes synchronous broadcast, whereas our protocols work even if parties communicate over an asynchronous, point-to-point network.

2 Model and Definitions

We denote the security parameter by k. Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be a function which may take negative values. We say ϵ is *negligible* if for all $c > 0$ there is a $k_c > 0$ such that $\epsilon(k) < 1/k^c$ for all $k > k_c$, and let negl denote a generic negligible function. We say ϵ is *noticeable* if there exist c, k_c such that $\epsilon(k) > 1/k^c$ for all $k > k_c$. Note that it is possible for ϵ to be neither negligible nor noticeable.

We define our model and then describe the game-theoretic concepts used. Even readers familiar with prior work in this area should skim the next few sections, since we formalize certain aspects of the problem slightly differently from prior work, and define new equilibrium notions.

2.1 Secret Sharing and Players' Utilities

A *t-out-of-n secret-sharing scheme for domain $\mathcal{S}$* (with $|\mathcal{S}| > 1$) is a two-phase protocol carried out by a dealer D and a set of n parties $P_1, \ldots, P_n$. In the first phase (the *sharing phase*), the dealer chooses a secret $s \in \mathcal{S}$. Based on this secret and a security parameter 1^k, the dealer generates shares $s_1, \ldots, s_n$ and gives s_i to player P_i. In the second phase (the *reconstruction phase*), some set I of $t^* \geq t$ active parties jointly reconstruct s. We impose the following requirements:

Secrecy: The shares of any $t-1$ parties reveal nothing about s, in a computational sense. Formally, for any $s_0, s_1 \in \mathcal{S}$ and any $i_1, \ldots, i_{t-1}$ the following are computationally indistinguishable:

$$\left\{(s_1, \ldots, s_n) \leftarrow D(1^k, s_0) : (s_{i_1}, \ldots, s_{i_{t-1}})\right\} \quad \text{and} \quad \left\{(s_1, \ldots, s_n) \leftarrow D(1^k, s_1) : (s_{i_1}, \ldots, s_{i_{t-1}})\right\}.$$

Correctness: For any set I of $t^* \geq t$ parties who run the reconstruction phase honestly, the correct secret s will be reconstructed, except possibly with probability negligible in k.

The above views parties as either malicious or honest. To model *rationality*, we define players' utilities. Given a set I of $t^* \geq t$ parties active during the reconstruction phase, let the outcome o of the reconstruction phase be a vector of length t^* with $o_i = 1$ iff the output of P_i is equal to the initial secret s (i.e., P_i "learned the secret"). We consider a party to have learned the secret s if and only if it outputs s, and do not care whether that party "really knows" the secret or not. In particular, a party who outputs a random value in $\mathcal{S}$ without running the reconstruction phase at all "learns" the secret with probability $1/|\mathcal{S}|$. We model the problem this way for two reasons:

1. Our formulation lets us model a player learning *partial* information about the secret, something not reflected in prior work. In particular, partial information that increases the probability with which a party outputs the correct secret increases that party's expected utility.
2. It is difficult, in general, to formally model what it means for a party to "really" learn the secret, especially when considering arbitrary protocols and behaviors. In contrast, in our definition it is easy to tell whether a player learns the secret by just looking at their output. Our notion also appears better suited for a computational setting, where a party might "know" the secret from an information-theoretic point of view, yet be unable to output it.

Let $\mu_i(o)$ be the utility of player P_i for the outcome o. Following [12] and most subsequent work (an exception is [4]), we make the following assumptions about the utility functions of the players:

- If $o_i > o'_i$, then $\mu_i(o) > \mu_i(o')$.
- If $o_i = o'_i$ and $\sum_i o_i < \sum_i o'_i$, then $\mu_i(o) > \mu_i(o')$.

That is, player P_i first prefers outcomes in which he learns the secret; otherwise, P_i prefers strategies in which the fewest number of other players learn the secret. For simplicity, in our analysis we distinguish three cases, described from the point of view of P_i (though we stress that we could also work with utilities satisfying the more general constraints above):

1. If o is an outcome in which P_i learns the secret and no other player does, then $\mu_i(o) \stackrel{\text{def}}{=} U^+$.
2. If o is an outcome in which P_i learns the secret and at least one other player does also, then $\mu_i(o) \stackrel{\text{def}}{=} U$.
3. If o is an outcome in which P_i does not learn the secret, then $\mu_i(o) \stackrel{\text{def}}{=} U^-$.

(Note that U^+, U, U^- are treated as fixed constants, independent of the security parameter.) Our conditions impose $U^+ > U > U^-$. Define

$$U_{\text{random}} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{S}|} \cdot U^+ + \left(1 - \frac{1}{|\mathcal{S}|}\right) \cdot U^- \quad ; \tag{1}$$

this is the expected utility of a party who outputs a random guess for the secret (assuming other parties abort without any output, or with the wrong output). We will also assume that $U > U_{\text{random}}$; otherwise, players have (almost) no incentive to run the reconstruction phase at all.

Strategies in our context refer to probabilistic polynomial-time interactive Turing machines. Given a vector of strategies $\vec{\sigma}$ for a set of t^* parties active in the reconstruction phase, we let $U_i(\vec{\sigma})$ denote the expected utility of P_i. (Note that the expected utility is a function of the security parameter k.) This expectation is taken over the initial choice of s (which we will always assume to be uniform), the dealer's randomness, and the randomness of the players' strategies. Following standard game-theoretic notation, define $\vec{\sigma}_{-i} \stackrel{\text{def}}{=} (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_{t^*})$ and $(\sigma'_i, \vec{\sigma}_{-i}) \stackrel{\text{def}}{=} (\sigma_1, \ldots, \sigma_{i-1}, \sigma'_i, \sigma_{i+1}, \ldots, \sigma_{t^*})$; that is, $(\sigma'_i, \vec{\sigma}_{-i})$ denotes the strategy vector $\vec{\sigma}$ with P_i's strategy changed to σ'_i.

2.2 Notions of Game-Theoretic Equilibria: A Discussion

The starting point for any discussion of game-theoretic equilibria is the *Nash equilibrium*. Roughly speaking, a protocol induces a Nash equilibrium if no party gains any advantage by deviating from the protocol, as long as all other parties follow the protocol. (In a *computational* Nash equilibrium, no *efficient* deviation confers any advantage.) As observed by Halpern and Teague [12], however, the Nash equilibrium concept is too weak for rational secret sharing. Halpern and Teague suggest, instead, to design protocols that induce a Nash equilibrium *surviving iterated deletion of weakly dominated strategies*; this notion was used in subsequent work of [10, 23, 1].

The notion of surviving iterated deletion, though, is also problematic in several respects. Kol and Naor [18] show a secret-sharing protocol that is "intuitively bad" yet satisfies the definition because no strategy weakly dominates any other: for any strategies σ, σ', there exist (contrived) strategies of the remaining players for which σ is the better strategy, and vice versa. (See [15, 16] for other arguments against this notion.) Also, a notion of surviving iterated deletion taking *computational* issues into account has not yet been defined (and doing so appears difficult).

Motivated by these drawbacks (and more), researchers have recently proposed other strengthenings of the Nash equilibrium concept [15, 17, 18, 4]. Kol and Naor define the notions of *resistance to backward induction* [17], *everlasting equilibrium*, and *strict Nash equilibrium* [18]. The latter two notions are defined in an information-theoretic sense, and are overly conservative in that they rule out protocols using cryptography; indeed, Kol and Naor state [18] that these equilibrium notions should be considered *sufficient* but not *necessary*. Nevertheless, the notion of strict Nash equilibrium is appealing. A protocol is in Nash equilibrium if no deviations are advantageous; it is in *strict* Nash equilibrium if all deviations are *dis*advantageous. Put differently, in the case of a Nash equilibrium there is no incentive to deviate whereas in the case of a strict Nash equilibrium there is an incentive *not* to deviate.

Another advantage of strict Nash is that protocols satisfying this notion deter subliminal communication in the following sense: since *any* detectable deviation from the protocol results in lower utility (assuming other parties are following the protocol), a party who tries to use the messages of the protocol as a covert channel risks a loss in utility as long as there is some reasonable probability that other players are following the protocol. In fact, our protocols satisfy the following, stronger condition: at every point in the protocol, there is a *unique* legal message that a party can send. Our protocols thus rule out subliminal communication in a strong sense; this was an explicit goal in work such as [19, 21, 20, 3].

We propose here a *computational* version of strict Nash equilibrium. We believe our definition retains the intuitive appeal of strict Nash, while also meaningfully taking computational limitations into account (and thus enabling the use of cryptography).

We also define a computational notion of *stability with respect to trembles*. Intuitively, stability with respect to trembles models players' uncertainty about other parties' behavior, and guarantees that even if a party P_i believes that other parties might play some arbitrary strategy with small probability δ (but follow the protocol with probability $1-\delta$), there is still no better strategy for P_i than to follow the protocol. Our formulation of this notion follows the general suggestion of Katz [15], but we flesh out the (non-trivial) technical details. An alternate formulation (*trembling-hand perfect equilibrium*), with somewhat different motivation, is discussed in [26].

As should be clear, determining the "right" game-theoretic notions for rational secret sharing is the subject of ongoing research. We do not suggest that the definitions proposed here are the *only* ones to consider, but we do believe they contribute to our understanding of the problem.

2.3 Definitions of Game-Theoretic Equilibria

Here we focus on the two-party case; definitions for the multi-party case, for both single-player deviations and coalitions, are given in Appendix B. In this section, Π is a 2-out-of-2 secret-sharing scheme and σ_i (for $i \in \{1,2\}$) denotes the prescribed actions of P_i in the reconstruction phase. We first define the most basic equilibrium notion for secret sharing.

Definition 1 Π induces a *computational Nash equilibrium* if for any PPT strategy σ_1' of P_1 we have $U_1(\sigma_1', \sigma_2) \leq U_1(\sigma_1, \sigma_2) + \mathsf{negl}(k)$, and similarly for P_2. $\diamondsuit$

Our definitions of strict Nash and resistance to trembles require us to first define what it means to "follow a protocol". This is non-trivial since a *different* Turing machine ρ_1 might be "functionally identical" to the prescribed strategy σ_1 as far as the protocol is concerned: for example, ρ_1 may be the same as σ_1 except that it first performs some useless computation; the strategies may be identical except that ρ_1 uses pseudorandom coins instead of random coins; or, the two strategies may differ in the message(s) they send *after* the protocol ends. In any of these cases we would like to say that ρ_1 is essentially "the same" as σ_1. This motivates the following definition, stated for the case of a deviating P_1 (with an analogous definition for a deviating P_2):

Definition 2 Define the random variable view_2^{Π} as follows:

> P_1 and P_2 interact, following σ_1 and σ_2, respectively. Let trans denote the messages sent by P_1 *not including any messages sent by P_1 after it writes to its output tape*. Then view_2^{Π} includes the information given by the dealer to P_2, the random coins of P_2, and the (partial) transcript trans.

Fix a strategy ρ_1 and an algorithm T. Define the random variable $\mathsf{view}_2^{T,\rho_1}$ as follows:

> P_1 and P_2 interact, following ρ_1 and σ_2, respectively. Let trans denote the messages sent by P_1. Algorithm T, given the entire view of P_1, outputs an arbitrary *truncation* trans' of trans. (That is, it defines a cut-off point and deletes any messages sent after that point.) Then $\mathsf{view}_2^{T,\rho_1}$ includes the information given by the dealer to P_2, the random coins of P_2, and the (partial) transcript trans'.

Strategy ρ_1 *yields equivalent play with respect to* Π, denoted $\rho_1 \approx \Pi$, if there exists a PPT algorithm T such that for all PPT distinguishers D

$$\left|\Pr[D(1^k, \mathsf{view}_2^{T,\rho_1}) = 1] - \Pr[D(1^k, \mathsf{view}_2^{\Pi}) = 1]\right| \leq \mathsf{negl}(k). \qquad \diamondsuit$$

We write $\rho_1 \not\approx \Pi$ if ρ_1 does *not* yield equivalent play with respect to Π. Note that ρ_1 can yield equivalent play with respect to Π even if (1) it differs from the prescribed strategy when interacting with some *other* strategy σ_2' (we only care about the behavior of ρ_1 when the other party runs Π); (2) it differs from the prescribed strategy in the local computation or output; and (3) it differs from the prescribed strategy after P_1 computes its output. This last point models the fact that we cannot force P_1 to send "correct" messages once, as far as P_1 is concerned, the protocol is finished.

We now define the notion that *detectable* deviations from the protocol *decrease* a player's utility.

Definition 3 Π induces a *computational strict Nash equilibrium* if

1. Π induces a computational Nash equilibrium;
2. For any PPT strategy σ_1' with $\sigma_1' \not\approx \Pi$, there is a $c > 0$ such that $U_1(\sigma_1, \sigma_2) \geq U_1(\sigma_1', \sigma_2) + 1/k^c$ for infinitely many values of k (with an analogous requirement for a deviating P_2). $\diamondsuit$

We next turn to defining stability with respect to trembles. We say that ρ_i *is δ-close to σ_i* if ρ_i takes the following form: with probability $1 - \delta$ party P_i plays σ_i, while with probability δ it follows an arbitrary PPT strategy σ_i'. (In this case, we refer to σ_i' as the *residual strategy* of ρ_i.) The notion of δ-closeness is meant to model a situation in which P_i plays σ_i "almost always," but with some (small) probability plays some other arbitrary strategy.

Intuitively, a pair of strategies (σ_1, σ_2) is *stable with respect to trembles* if σ_1 (resp., σ_2) remains a best response even if the other party plays a strategy other than σ_2 (resp., σ_1) with some small (but noticeable) probability δ. As in the case of strict Nash equilibrium, this notion is difficult to define formally because of the possibility that one party can do better (in case the other deviates) by performing some *local* computation.[1] Our definition essentially requires that this is the *only* way for either party to do better and so, in particular, each party will (at least outwardly) continue to follow the protocol until the other deviates. Moreover, any (polynomial-time) local computation performed by either party is of no benefit as long as the other party follows the protocol.

Definition 4 Π induces a *computational Nash equilibrium that is stable with respect to trembles* if

1. Π induces a computational Nash equilibrium;

2. There is a noticeable function δ such that for any PPT strategy ρ_2 that is δ-close to σ_2, and any PPT strategy ρ_1, there exists a PPT strategy $\sigma_1' \approx \Pi$ such that $U_1(\rho_1, \rho_2) \leq U_1(\sigma_1', \rho_2) + \mathsf{negl}(k)$ (with an analogous requirement for the case of deviations by P_2). $\diamondsuit$

Stated differently, even if a party P_i believes that the other party might play a different strategy with some small probability δ, there is still no better strategy for P_i than to outwardly follow the protocol[2] (while possibly performing some additional local computation). Moreover, if Π induces a computational Nash equilibrium then any (polynomial-time) local computation performed by P_i will not help as long as the other party follows the protocol.

2.4 Verifiable Random Functions (VRFs)

A VRF is a keyed function whose output is "random-looking" but can still be verified as correct, given an associated proof. The notion was introduced by Micali, Rabin, and Vadhan [24], and various constructions in the standard model are known [24, 6, 22, 8]. The definition we use is stronger than the "standard" one in that it includes a uniqueness requirement on the *proof* as well, but the constructions of [6, 8] achieve it. (Also, we use VRFs only as a stepping stone to our construction based on trapdoor permutations.)

Definition 5 A *verifiable random function* (VRF) with range $\mathcal{R} = \{\mathcal{R}_k\}$ is a tuple of probabilistic polynomial-time algorithms (Gen, Eval, Prove, Vrfy) such that the following hold:

Correctness: For all k, any (pk, sk) output by $\mathsf{Gen}(1^k)$, the algorithm Eval_{sk} maps k-bit inputs to the set $\mathcal{R}_k$. Furthermore, for any $x \in \{0,1\}^k$ we have $\mathsf{Vrfy}_{pk}(x, \mathsf{Eval}_{sk}(x), \mathsf{Prove}_{sk}(x)) = 1$.

Verifiability: For all (pk, sk) output by $\mathsf{Gen}(1^k)$, there does not exist a tuple (x, y, y', π, π') with $y \neq y'$ and $\mathsf{Vrfy}_{pk}(x, y, \pi) = 1 = \mathsf{Vrfy}_{pk}(x, y', \pi')$.

Unique proofs: For all (pk, sk) output by $\mathsf{Gen}(1^k)$, there does not exist a tuple (x, y, π, π') with $\pi \neq \pi'$ and $\mathsf{Vrfy}_{pk}(x, y, \pi) = 1 = \mathsf{Vrfy}_{pk}(x, y, \pi')$.

Pseudorandomness: Consider the following experiment involving an adversary $\mathcal{A}$:

1. Generate $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$ and give pk to $\mathcal{A}$. Then $\mathcal{A}$ adaptively queries a sequence of strings $x_1, \ldots, x_\ell \in \{0,1\}^k$ and is given $y_i = \mathsf{Eval}_{sk}(x_i)$ and $\pi_i = \mathsf{Prove}_{sk}(x_i)$ in response to each such query x_i.
2. $\mathcal{A}$ outputs a string $x \in \{0,1\}^k$ subject to the restriction $x \notin \{x_1, \ldots, x_\ell\}$.

[1] As a trivial example, consider the case where with probability δ one party just sends its share to the other.

[2] Specifically, for any strategy ρ_i that does *not* yield equivalent play w.r.t. Π, there exists a strategy σ_i' that *does* yield equivalent play w.r.t. Π and performs essentially as well.

3. A random bit $b \leftarrow \{0,1\}$ is chosen. If $b = 0$ then $\mathcal{A}$ is given $y = \mathsf{Eval}_{sk}(x)$; if $b = 1$ then $\mathcal{A}$ is given a random $y \leftarrow \mathcal{R}_k$.
4. $\mathcal{A}$ makes more queries as in step 2, as long as none of these queries is equal to x.
5. At the end of the experiment, $\mathcal{A}$ outputs a bit b'. We say $\mathcal{A}$ *succeeds* if $b' = b$.

We require that for any PPT adversary $\mathcal{A}$, the success probability of $\mathcal{A}$ is $\frac{1}{2} + \mathsf{negl}(k)$. $\diamondsuit$

3 Rational Secret Sharing: The 2-out-of-2 Case

Let $\mathcal{S} = \{0,1\}^\ell$ be the domain of the secret, where ℓ may depend on the security parameter k. Let $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Prove}, \mathsf{Vrfy})$ be a VRF with range $\{0,1\}^\ell$, and let $(\mathsf{Gen}', \mathsf{Eval}', \mathsf{Prove}', \mathsf{Vrfy}')$ be a VRF with range $\{0,1\}^k$. Protocol Π is defined as follows:

Sharing phase: Let s denote the secret. The dealer chooses an integer $i^* \in \mathbb{N}$ according to a geometric distribution with parameter β, where β is a constant that depends on the players' utilities but is independent of the security parameter; we discuss how to set β below. We assume $i^* < 2^k - 1$ since this occurs with all but negligible probability. (Technically, if $i^* \geq 2^k - 1$ the dealer can just send a special error message to each party.)

The dealer computes $(pk_1, sk_1), (pk_2, sk_2) \leftarrow \mathsf{Gen}(1^k)$ and $(pk'_1, sk'_1), (pk'_2, sk'_2) \leftarrow \mathsf{Gen}'(1^k)$, and:

- $\mathsf{share}_1 := \mathsf{Eval}_{sk_2}(i^*) \oplus s$ and $\mathsf{share}_2 := \mathsf{Eval}_{sk_1}(i^*) \oplus s$;
- $\mathsf{signal}_1 := \mathsf{Eval}'_{sk'_2}(i^* + 1)$ and $\mathsf{signal}_2 := \mathsf{Eval}'_{sk'_1}(i^* + 1)$.

Finally, the dealer gives to P_1 the values $(sk_1, sk'_1, pk_2, pk'_2, \mathsf{share}_1, \mathsf{signal}_1)$, and gives to P_2 the values $(sk_2, sk'_2, pk_1, pk'_1, \mathsf{share}_2, \mathsf{signal}_2)$.

Reconstruction phase

At the outset of this phase, P_1 chooses $s_1^{(0)}$ uniformly from $\mathcal{S} = \{0,1\}^\ell$ and P_2 chooses $s_2^{(0)}$ the same way. Then in each iteration $i = 1, \ldots$, the parties do the following:

(P_2 sends message to P_1:) P_2 computes $y_2^{(i)} := \mathsf{Eval}_{sk_2}(i)$, $\pi_2^{(i)} := \mathsf{Prove}_{sk_2}(i)$ and $z_2^{(i)} := \mathsf{Eval}'_{sk'_2}(i)$, $\bar{\pi}_2^{(i)} := \mathsf{Prove}'_{sk'_2}(i)$. It sends $(y_2^{(i)}, \pi_2^{(i)}, z_2^{(i)}, \bar{\pi}_2^{(i)})$ to P_1.

(P_1 receives message from P_2:) P_1 receives $(y_2^{(i)}, \pi_2^{(i)}, z_2^{(i)}, \bar{\pi}_2^{(i)})$ from P_2. If P_2 does not send anything, or if $\mathsf{Vrfy}_{pk_2}(i, y_2^{(i)}, \pi_2^{(i)}) = 0$ or $\mathsf{Vrfy}'_{pk'_2}(i, z_2^{(i)}, \bar{\pi}_2^{(i)}) = 0$, then P_1 outputs $s_1^{(i-1)}$ and halts.

If $\mathsf{signal}_1 \stackrel{?}{=} z_2^{(i)}$ then P_1 outputs $s_1^{(i-1)}$, send its iteration-i message to P_2 (see below), and halts. Otherwise, it sets $s_1^{(i)} := \mathsf{share}_1 \oplus y_2^{(i)}$ and continues.

(P_1 sends message to P_2:) P_1 computes $y_1^{(i)} := \mathsf{Eval}_{sk_1}(i)$, $\pi_1^{(i)} := \mathsf{Prove}_{sk_1}(i)$ and $z_1^{(i)} := \mathsf{Eval}'_{sk'_1}(i)$, $\bar{\pi}_1^{(i)} := \mathsf{Prove}'_{sk'_1}(i)$. It sends $(y_1^{(i)}, \pi_1^{(i)}, z_1^{(i)}, \bar{\pi}_1^{(i)})$ to P_2.

(P_2 receives message from P_1:) P_2 receives $(y_1^{(i)}, \pi_1^{(i)}, z_1^{(i)}, \bar{\pi}_1^{(i)})$ from P_1. If P_1 does not send anything, or if $\mathsf{Vrfy}_{pk_1}(i, y_1^{(i)}, \pi_1^{(i)}) = 0$ or $\mathsf{Vrfy}'_{pk'_1}(i, z_1^{(i)}, \bar{\pi}_1^{(i)}) = 0$, then P_2 outputs $s_2^{(i-1)}$ and halts.

If $\mathsf{signal}_2 \stackrel{?}{=} z_1^{(i)}$ then P_2 outputs $s_2^{(i-1)}$ and halts. Otherwise, it sets $s_2^{(i)} := \mathsf{share}_2 \oplus y_1^{(i)}$ and continues.

Figure 1: The reconstruction phase of secret-sharing protocol Π.

Reconstruction phase: A high-level overview of the protocol was given in Section 1.1, and we give the formal specification in Figure 1. The reconstruction phase proceeds in a series of iterations, where each iteration consists of one message sent by each party. Although these messages could be sent at the same time (since they do not depend on each other), we do not want to assume simultaneous communication and therefore simply require P_2 to communicate first in each iteration. (If one were willing to assume simultaneous channels then the protocol could be simplified by having P_2 send $\mathsf{Eval}'_{sk'_2}(i+1)$ at the same time as $\mathsf{Eval}_{sk_2}(i)$, and similarly for P_1.)

We give some intuition as to why the reconstruction phase of Π is a computational Nash equilibrium for an appropriate choice of β. Assume P_2 follows the protocol, and consider possible deviations by P_1. (Deviations by P_2 are even easier to analyze since P_2 goes first in every iteration.) P_1 can abort in iteration $i = i^* + 1$ (i.e., as soon as it receives $z_2^{(i)} = \mathsf{signal}_1$), or it can abort in some iteration $i < i^* + 1$. In the first case P_1 "knows" that it learned the dealer's secret in the preceding iteration (that is, in iteration i^*) and can thus output the correct secret; however, P_2 will output $s_2^{(i^*)} = s$ and so learns the secret as well. So P_1 does not increase its utility beyond what it would achieve by following the protocol. In the second case, when P_1 aborts in some iteration $i < i^* + 1$, the best strategy P_1 can adopt is to output $s_1^{(i)}$ and hope that $i = i^*$. The expected utility that P_1 obtains by following this strategy can be calculated as follows:

- P_1 aborts exactly in iteration $i = i^*$ with probability β. In this case, P_1 gets utility at most U^+.
- When $i < i^*$, player P_1 has "no information" about s and so the best it can do is guess. The expected utility of P_1 in this case is thus at most U_{random} (cf. Equation (1)).

Putting everything together, the expected utility of P_1 following this strategy is at most

$$\beta \times U^+ + (1 - \beta) \times U_{\text{random}} .$$

Since $U_{\text{random}} < U$ by assumption, it is possible to set β so that the entire expression above is strictly less than U; in that case, P_1 has no incentive to deviate.

That Π induces a *strict* computational Nash equilibrium (that is also stable with respect to trembles) follows from the fact that there is always a *unique* valid message that a party can send; anything else is treated as an abort. A proof of the following appears in Appendix A.

Theorem 1 *Let $\beta > 0$ be such that $U > \beta \cdot U^+ + (1-\beta) \cdot U_{\text{random}}$. Then Π induces a computational strict Nash equilibrium that is stable with respect to trembles.*

3.1 Using Trapdoor Permutations Instead of VRFs

The protocol from the previous section can be adapted easily to use trapdoor permutations instead of VRFs. The key observation is that the VRFs in the previous protocol are used only in a very specific way: they applied *sequentially* to values $1, 2, \ldots$. One can therefore use a trapdoor permutation f with associated hardcore bit h to instantiate the VRF in our scheme in the following way: The public key is a description of f along with a random element y in the domain of f; the secret key is the trapdoor enabling inversion of f. In iteration i, the "evaluation" of the VRF on input i is the ℓ-bit sequence

$$h\left(f^{-(i-1)\ell-1}(y)\right), h\left(f^{-(i-1)\ell-2}(y)\right), \ldots, h\left(f^{-(i-1)\ell-\ell}(y)\right) ,$$

and the "proof" is $\pi_i = f^{-(i-1)\ell-\ell}(y)$. Verification can be done using the original point y, and can also be done in time independent of i by using π_{i-1} (namely, by checking that $f^{\ell}(\pi_i) = \pi_{i-1}$), assuming π_{i-1} has already been verified.

The key point is that the essential properties we need still hold: verifiability and uniqueness of proofs are easy to see, and pseudorandomness still holds with respect to a modified game where the adversary queries $\mathsf{Eval}_{sk}(1), \ldots, \mathsf{Eval}_{sk}(i)$ and then has to guess whether it is given $\mathsf{Eval}_{sk}(i+1)$ or a random string. We omit further details.

4 Rational Secret Sharing: The t-out-of-n Case

In this section we describe extensions of our protocol to the t-out-of-n case, where we consider deviations by coalitions of up to $t-1$ parties. Formal definitions of game-theoretic notions in the multi-player setting, both for the case of single-player deviations as well as coalitions, are fairly straightforward adaptations of the definitions from Section 2.3 and are given in Appendix B.

In describing our protocols we use VRFs for notational simplicity, but all the protocols given here can be instantiated using trapdoor permutations as described in Section 3.1.

A protocol for "exactly t-out-of-n" secret sharing. We begin by describing a protocol $\Pi_{t,n}$ for t-out-of-n secret sharing that is resilient to coalitions of up to $t-1$ parties under the assumption that *exactly* t parties are active during the reconstruction phase. (We also require that the coalition be a subset of the active parties.) For now, we assume communication over a synchronous (but not simultaneous) point-to-point network.

As in the 2-out-of-2 case, every party is associated with two keys for a VRF. The dealer chooses an iteration r^* according to a geometric distribution, and also chooses two random $(t-1)$-degree polynomials G, H (over some finite field) such that $G(0) = s$ and $H(0) = 0$. Each party receives *blinded* versions of all n points $\{G(j), H(j)\}_{j=1}^{n}$: each $G(j)$ is blinded by the value of P_j's VRF on the input r^*, and each $H(j)$ is blinded by the value of P_j's VRF on the input r^*+1. In each iteration r, each party is supposed to send to all other parties the value of their VRFs evaluated on the current iteration number r; once this is done, every party can interpolate a polynomial to obtain candidate values for $G(0)$ and $H(0)$. When $H(0) = 0$ parties know the protocol is over, and output the $G(0)$ value reconstructed in the *previous* iteration. See Figure 2 for details.

Theorem 2 *Let $\beta > 0$ be such that $U > \beta \cdot U^+ + (1-\beta) \cdot U_{\mathrm{random}}$. Then $\Pi_{t,n}$ induces a $(t-1)$-resilient computational strict Nash equilibrium that is stable with respect to trembles, as long as exactly t parties are active during the reconstruction phase.*

A proof is exactly analogous to the proof of Theorem 1.

Handling the general case. The prior solution assumes exactly t parties are active during reconstruction. If $t^* > t$ parties are active, the "natural" implementation of the protocol — where the lowest-indexed t parties run $\Pi_{t,n}$ and all other parties remain silent — is not a $(t-1)$-resilient computational Nash equilibrium. To see why, let the active parties be $I = \{1, \ldots t+1\}$ and let $\mathcal{C} = \{3, \ldots, t+1\}$ be a coalition of $t-1$ parties. In each iteration r, as soon as P_1 and P_2 send their values the parties in $\mathcal{C}$ can compute $t+1$ points $\{g_j^{(r)}\}_{j \in I}$. Because of the way these points are constructed, they are guaranteed to lie on a $(t-1)$-degree polynomial when $r = r^*$, but are unlikely to lie on a $(t-1)$-degree polynomial when $r < r^*$. This gives the parties in $\mathcal{C}$ a way to

Sharing Phase

To share a secret $s \in \{0,1\}^\ell$, the dealer does the following:

- Choose $r^* \in \mathbb{N}$ according to a geometric distribution with parameter β.
- Generate[a] $(pk_1, sk_1), \ldots, (pk_n, sk_n) \leftarrow \mathsf{Gen}(1^k)$ and $(pk'_1, sk'_1), \ldots, (pk'_n, sk'_n) \leftarrow \mathsf{Gen}(1^k)$.
- Choose random $(t-1)$-degree polynomials $G \in \mathbb{F}_{2^\ell}[x]$ and $H \in \mathbb{F}_{2^k}[x]$ such that $G(0) = s$ and $H(0) = 0$.
- Send sk_i, sk'_i to player P_i, and send to all parties the following values:
  1. $\{(pk_j, pk'_j)\}_{1 \leq j \leq n}$
  2. $\{g_j := G(j) \oplus \mathsf{Eval}_{sk_j}(r^*)\}_{1 \leq j \leq n}$
  3. $\{h_j := H(j) \oplus \mathsf{Eval}'_{sk'_j}(r^* + 1)\}_{1 \leq j \leq n}$

Reconstruction Phase

Let I be the set of indices of the t active players. Each party P_i (for $i \in I$) chooses $s_i^{(0)}$ uniformly from $\{0,1\}^\ell$. In each iteration $r = 1, \ldots,$ the parties do:

- For all $i \in I$ (in ascending order), P_i sends the following to all players:

$$\left(y_i^{(r)} := \mathsf{Eval}_{sk_i}(r),\ z_i^{(r)} := \mathsf{Eval}'_{sk'_i}(r),\ \mathsf{Prove}_{sk_i}(r),\ \mathsf{Prove}'_{sk'_i}(r) \right).$$

- If a party P_i receives an incorrect proof (or nothing) from any other party P_j, then P_i terminates and outputs $s_i^{(r-1)}$. Otherwise:
  - P_i sets $h_j^{(r)} := h_j \oplus z_j^{(r)}$ for all $j \in I$, and interpolates a degree-$(t-1)$ polynomial $H^{(r)}$ through the t points $\{h_j^{(r)}\}_{j \in I}$. If $H^{(r)}(0) \stackrel{?}{=} 0$ then P_i outputs $s_i^{(r-1)}$ immediately, and terminates after sending its current-iteration message.
  - Otherwise, P_i compute $s_i^{(r)}$ as follows: set $g_j^{(r)} := g_j \oplus y_j^{(r)}$ for all $j \in I$. Interpolate a degree-$(t-1)$ polynomial $G^{(r)}$ through the points $\{g_j^{(r)}\}_{j \in I}$, and set $s_i^{(r)} := G^{(r)}(0)$.

[a] Gen outputs VRF keys with range $\{0,1\}^\ell$, and Gen' outputs VRF keys with range $\{0,1\}^k$.

Figure 2: Protocol $\Pi_{t,n}$ for "exactly t-out-of-n" secret sharing.

determine r^* as soon as that iteration is reached, at which point they can abort and output the secret while preventing P_1 and P_2 from doing the same.

Fortunately, a simple modification works: simply have the dealer run independent instances $\Pi_{t,n}, \Pi_{t+1,n}, \ldots, \Pi_{n,n}$; in the reconstruction phase, the parties run $\Pi_{t^*,n}$ where t^* denotes the number of active players. It follows as an easy corollary of Theorem 2 that this induces a $(t-1)$-resilient computational strict Nash equilibrium (that is also stable with respect to trembles) regardless of how many parties are active during the reconstruction phase. (As in previous work, we only consider coalitions that are subsets of the parties who are active during reconstruction. The protocol is no longer a computational Nash equilibrium if this is not the case.[3])

[3] This case can be addressed, however, by having the dealer run independent instances of $\Pi_{t,n}$ for all $\binom{n}{t}$ subsets of size t; to reconstruct, the t lowest-indexed active players run the instance corresponding to their subset while the remaining active players are silent. This is only efficient for small values of t.

Asynchronous networks. Our protocol $\Pi_{t,n}$ can be adapted to work even when the parties communicate over an *asynchronous* point-to-point network. (Here messages can be delayed arbitrarily and delivered out of order, but any message that is sent is eventually delivered.) In the asynchronous case, parties cannot distinguish an abort from a delayed message and so we modify the protocol as follows: each party proceeds to the next iteration r as soon as it receives $t-1$ valid messages from the previous iteration, and only halts if it receives an invalid message from someone. More formal treatment of the asynchronous case, including a discussion of definitions in this setting and a proof for the preceding protocol, is deferred to Appendix C.

As before, we can handle the general case by having the dealer run independent instances of the "exactly t^*-out-of-n" protocol just described for all values of $t^* \in \{t, \ldots, n\}$. (We continue to restrict attention to coalitions that consist only of active players.)

References

[1] I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *25th ACM Symposium Annual on Principles of Distributed Computing*, pages 53–62. ACM Press, 2006.

[2] J. Alwen, J. Katz, Y. Lindell, G. Persiano, A. Shelat, and I. Visconti. Collusion-free multiparty computation in the mediated model. In *Advances in Cryptology — Crypto 2009 (to appear)*, volume ??? of *LNCS*, pages ???–??? Springer, 2009.

[3] J. Alwen, A. Shelat, and I. Visconti. Collusion-free protocols in the mediated model. In *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 497–514. Springer, 2008.

[4] G. Asharov and Y. Lindell. Utility dependence in correct and fair rational secret sharing. In *Advances in Cryptology — Crypto 2009 (to appear)*, volume ??? of *LNCS*, pages ???–??? Springer, 2009.

[5] G. Blakley. Safeguarding cryptographic keys. *National Computer Conference*, 48:313–317, 1979.

[6] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *LNCS*, pages 1–17. Springer, 2003.

[7] Y. Dodis and T. Rabin. Cryptography and game theory. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, pages 181–207. Cambridge University Press, 2007.

[8] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.

[9] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 413–422. ACM Press, 2008.

[10] S. D. Gordon and J. Katz. Rational secret sharing, revisited. In *5th Intl. Conf. on Security and Cryptography for Networks (SCN)*, volume 4116 of *LNCS*, pages 229–241. Springer, 2006.

[11] S. D. Gordon and J. Katz. Partial fairness in secure two-party computation, 2008. Available at `http://eprint.iacr.org/2008/206`.

[12] J. Halpern and V. Teague. Rational secret sharing and multiparty computation: Extended abstract. In *36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 623–632. ACM Press, 2004.

[13] S. Izmalkov, M. Lepinski, and S. Micali. Verifiably secure devices. In *5th Theory of Cryptography Conference — TCC 2008*, volume 4948 of *LNCS*, pages 273–301. Springer, 2008.

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

[15] J. Katz. Bridging game theory and cryptography: Recent results and future directions. In *5th Theory of Cryptography Conference — TCC 2008*, volume 4948 of *LNCS*, pages 251–272. Springer, 2008.

[16] J. Katz. Ruminations on defining rational MPC, 2008. Talk given at SSoRC, Bertinoro, Italy. Slides available at `http://www.daimi.au.dk/~jbn/SSoRC2008/program`.

[17] G. Kol and M. Naor. Cryptography and game theory: Designing protocols for exchanging information. In *5th Theory of Cryptography Conference — TCC 2008*, volume 4948 of *LNCS*, pages 320–339. Springer, 2008.

[18] G. Kol and M. Naor. Games for exchanging information. In *40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 423–432. ACM Press, 2008.

[19] M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely fair SFE and coalition-safe cheap talk. In *23rd ACM Symposium Annual on Principles of Distributed Computing*, pages 1–10. ACM Press, 2004.

[20] M. Lepinski, S. Micali, and A. Shelat. Collusion-free protocols. In *37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 543–552. ACM Press, 2005.

[21] M. Lepinski, S. Micali, and A. Shelat. Fair-zero knowledge. In *2nd Theory of Cryptography Conference — TCC 2005*, volume 3378 of *LNCS*, pages 245–263. Springer, 2005.

[22] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Advances in Cryptology — Crypto 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, 2002.

[23] A. Lysyanskaya and N. Triandopoulos. Rationality and adversarial behavior in multi-party computation. In *Advances in Cryptology — Crypto 2006*, volume 4117 of *LNCS*, pages 180–197. Springer, 2006.

[24] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 120–130. IEEE, 1999.

[25] S. Miclai and A. Shelat. Truly rational secret sharing. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 54–71. Springer, 2009.

[26] S. J. Ong, D. Parkes, A. Rosen, and S. Vadhan. Fairness with an honest minority and a rational majority. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 36–53. Springer, 2009.

[27] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

A Proof of Theorem 1

We first show that Π is a valid secret-sharing scheme. Secrecy follows from the proof that the reconstruction phase is a computational Nash equilibrium, below, for if secrecy did not hold then computing the secret locally and not participating in the reconstruction phase at all would be a profitable deviation. We therefore focus on correctness. Assuming both parties run the protocol honestly, the correct secret is reconstructed unless:

- $i^* \geq 2^k - 1$.
- For some $i < i^* + 1$, either $\mathsf{signal}_1 = \mathsf{Eval}'_{sk'_2}(i)$ or $\mathsf{signal}_2 = \mathsf{Eval}'_{sk'_1}(i)$.

The first event occurs with negligible probability. Pseudorandomness of the VRF, along with the fact that $i^* \leq k$ with all but negligible probability, easily imply that the latter two events happen with only negligible probability as well.

We next prove that Π induces a computational Nash equilibrium. Assume P_2 follows the strategy σ_2 prescribed by the protocol, and let σ'_1 denote any PPT strategy followed by P_1. (The other case, where P_1 follows the protocol and we look at deviations by P_2, follows similarly with an even simpler proof.) In a given execution of the reconstruction phase, let i denote the iteration in which P_1 aborts (where an incorrect message is viewed as an abort); if P_1 never aborts then set $i = \infty$. Let early be the event that $i < i^*$; let exact be the event that $i = i^*$; and let late be the event that $i > i^*$. Let $\mathsf{correct}$ be the event that P_1 outputs the correct secret s. We will consider the probabilities of these events in two experiments: the experiment defined by running the actual secret-sharing scheme, and a second experiment where P_1 is given $\mathsf{share}_1, \mathsf{signal}_1$ chosen uniformly at random from the appropriate ranges. Probabilities in the first experiment will be denoted by $\Pr_{\text{real}}[\cdot]$, and probabilities in the second experiment will be denoted by $\Pr_{\text{ideal}}[\cdot]$. We have

$$\begin{aligned} & U_1(\sigma'_1, \sigma_2) \\ \leq \quad & U^+ \cdot \Pr_{\text{real}}[\mathsf{exact}] + U^+ \cdot \Pr_{\text{real}}[\mathsf{correct} \wedge \mathsf{early}] + U^- \cdot \Pr_{\text{real}}[\overline{\mathsf{correct}} \wedge \mathsf{early}] + U \cdot \Pr_{\text{real}}[\mathsf{late}], \end{aligned} \tag{2}$$

using the fact (as discussed in the intuition preceding the theorem) that whenever late occurs P_2 outputs the correct secret s. Since when both parties follow the protocol P_1 gets utility U, we need to show that there exists a negligible function ϵ such that $U_1(\sigma'_1, \sigma_2) \leq U + \epsilon(k)$.

The next claim follows easily from the pseudorandomness of the VRFs.

Claim 1 *There exists a negligible function ϵ such that*

$$\begin{aligned} |\Pr_{\text{real}}[\mathsf{exact}] - \Pr_{\text{ideal}}[\mathsf{exact}]| &\leq \epsilon(k) \\ |\Pr_{\text{real}}[\mathsf{late}] - \Pr_{\text{ideal}}[\mathsf{late}]| &\leq \epsilon(k) \\ |\Pr_{\text{real}}[\mathsf{correct} \wedge \mathsf{early}] - \Pr_{\text{ideal}}[\mathsf{correct} \wedge \mathsf{early}]| &\leq \epsilon(k) \\ |\Pr_{\text{real}}[\overline{\mathsf{correct}} \wedge \mathsf{early}] - \Pr_{\text{ideal}}[\overline{\mathsf{correct}} \wedge \mathsf{early}]| &\leq \epsilon(k). \end{aligned}$$

Define

$$U_{\text{ideal}} \stackrel{\text{def}}{=} U^+ \cdot \Pr_{\text{ideal}}[\mathsf{exact}] + U^+ \cdot \Pr_{\text{ideal}}[\mathsf{correct} \wedge \mathsf{early}] + U^- \cdot \Pr_{\text{ideal}}[\overline{\mathsf{correct}} \wedge \mathsf{early}] + U \cdot \Pr_{\text{ideal}}[\mathsf{late}].$$

Claim 1 shows that $|U_1(\sigma_1', \sigma_2) - U_{\text{ideal}}| \leq \epsilon(k)$ for some negligible function ϵ. It remains to bound U_{ideal}. Let $\mathsf{abort} = \mathsf{exact} \vee \mathsf{early}$, so that abort is the event that P_1 aborts before iteration $i^* + 1$. Information-theoretically, we have

$$\Pr_{\text{ideal}}[\mathsf{exact} \mid \mathsf{abort}] = \beta \quad \text{and} \quad \Pr_{\text{ideal}}[\mathsf{correct} \mid \mathsf{early}] = \frac{1}{|\mathcal{S}|};$$

therefore,

$$\begin{aligned} &U_{\text{ideal}} \\ &= U^+ \cdot \Big(\Pr_{\text{ideal}}[\mathsf{exact} \mid \mathsf{abort}] + \Pr_{\text{ideal}}[\mathsf{correct} \mid \mathsf{early}] \cdot \Pr_{\text{ideal}}[\mathsf{early} \mid \mathsf{abort}]\Big) \cdot \Pr_{\text{ideal}}[\mathsf{abort}] \\ &\quad + U^- \cdot \Pr_{\text{ideal}}[\overline{\mathsf{correct}} \mid \mathsf{early}] \cdot \Pr_{\text{ideal}}[\mathsf{early} \mid \mathsf{abort}] \cdot \Pr_{\text{ideal}}[\mathsf{abort}] + U \cdot \Pr_{\text{ideal}}[\mathsf{late}] \\ &= U^+ \cdot \left(\beta + \frac{1}{|\mathcal{S}|} \cdot (1-\beta)\right) \cdot \Pr_{\text{ideal}}[\mathsf{abort}] \\ &\quad + U^- \cdot \left(1 - \frac{1}{|\mathcal{S}|}\right)(1-\beta) \cdot \Pr_{\text{ideal}}[\mathsf{abort}] + U \cdot (1 - \Pr_{\text{ideal}}[\mathsf{abort}]) \\ &= U + \left(U^+ \cdot \left(\beta + \frac{1}{|\mathcal{S}|} \cdot (1-\beta)\right) + U^- \cdot \left(1 - \frac{1}{|\mathcal{S}|}\right)(1-\beta) - U\right) \cdot \Pr_{\text{ideal}}[\mathsf{abort}] \\ &= U + \big(\beta \cdot U^+ + (1-\beta) \cdot U_{\text{random}} - U\big) \cdot \Pr_{\text{ideal}}[\mathsf{abort}] \;\; \leq \;\; U \end{aligned} \tag{3}$$

using the fact that $\beta \cdot U^+ + (1-\beta) \cdot U_{\text{random}} - U < 0$. This completes the proof that Π induces a computational Nash equilibrium.

That Π induces a computational *strict* Nash equilibrium follows easily from the above analysis along with the fact that there is always a *unique* valid message each party can send. Specifically, say P_1 plays a strategy σ_1' with $\sigma_1' \not\approx \Pi$. This implies[4] that $\Pr_{\text{real}}[\mathsf{abort}] \geq 1/\mathsf{poly}(k)$ for infinitely many values of k. Claim 1 then shows that $\Pr_{\text{ideal}}[\mathsf{abort}] \geq 1/\mathsf{poly}(k)$ for infinitely many values of k, and so $U - U_{\text{ideal}} \geq 1/\mathsf{poly}(k)$ infinitely often as well (see Equation (3)). Since $|U_1(\sigma_1', \sigma_2) - U_{\text{ideal}}|$ is negligible, we conclude that $U - U_1(\sigma_1', \sigma_2) \geq 1/\mathsf{poly}(k)$ for infinitely many values of k, as required.

To complete the proof, we show that Π induces a computational Nash equilibrium stable with respect to trembles. The following table will be useful when reading the proof.

Strategy	Description
σ_1/σ_2	Prescribed by the protocol
ρ_2	A PPT strategy δ-close to σ_2; plays $\hat{\rho}_2$ with prob. δ, and σ_2 otherwise
$\hat{\rho}_2$	The PPT 'residual strategy' of ρ_2
ρ_1	Arbitrary PPT strategy for P_1
σ_1'	PPT strategy with $\sigma_1' \approx \Pi$; runs ρ_1 as a subroutine

[4] Recall that our definition of "equivalent play" (Definition 2) ignores deviations once P_1 could compute its output if it were following the prescribed protocol. So if $\sigma_1' \not\approx \Pi$, then the probability that P_1 aborts before iteration $i^* + 1$ is not negligible.

Let δ be a parameter we fix later. Let ρ_2 be any PPT strategy that is δ-close to σ_2, and let ρ_1 be an arbitrary PPT strategy for P_1. We show the existence of a PPT strategy σ_1' satisfying Definition 4. (Once again we focus on deviations by P_1, but the case of P_2 is analogous.) Strategy σ_1' is defined as follows:

1. Given input $(sk_1, sk_1', pk_2, pk_2', \mathsf{share}_1, \mathsf{signal}_1)$, run ρ_1 on this input. Set $\mathsf{aborted} := 0$.
2. In each iteration i:
   (a) Receive the iteration-i message m_i from P_2. If P_2 aborts, then set $\mathsf{aborted} := 1$.
   (b) Give m_i to ρ_1 and get in response some message m_i'.
   (c) If $\mathsf{aborted} = 1$ then forward m_i' to P_2; otherwise, compute the response $(y_1^{(i)}, \pi_1^{(i)}, z_1^{(i)}, \bar{\pi}_1^{(i)})$ as prescribed by Π and send that to P_2 instead.
3. If $\mathsf{aborted} = 0$ determine the output according to Π; otherwise, output whatever ρ_1 outputs.

When σ_1' interacts with σ_2, then $\mathsf{aborted}$ is never set to 1; thus, σ_1' yields equivalent play with respect to Π, and $U_1(\sigma_1', \sigma_2) = U_1(\sigma_1, \sigma_2) = U$. It remains to show that $U_1(\rho_1, \rho_2) \leq U_1(\sigma_1', \rho_2) + \mathsf{negl}(k)$. Let $\hat{\rho}_2$ denote the "residual strategy" of ρ_2; i.e., $\hat{\rho}_2$ is run only with probability δ by ρ_2. In an interaction where P_1 follows strategy ρ_1, let abort denote the event that ρ_1 aborts before P_2 aborts, and let $p_{\text{abort}}(\sigma)$ be the probability of abort when P_2 follows strategy σ. We first claim that the only "advantage" to P_1 of playing ρ_1 rather than σ_1' arises due to ρ_1 aborting first, i.e., due to the occurrence of abort:

Claim 2 $U_1(\rho_1, \hat{\rho}_2) - U_1(\sigma_1', \hat{\rho}_2) \leq p_{\text{abort}}(\hat{\rho}_2) \cdot (U^+ - U^-)$.

Proof Note that abort is well-defined in the interaction of σ_1' with $\hat{\rho}_2$, because σ_1' runs a copy of ρ_1 as a sub-routine. When abort does *not* occur, there are two possibilities: neither ρ_1 nor P_2 ever aborts, or P_2 aborts first. We consider these in turn:

- When neither ρ_1 nor P_2 aborts, the output of P_2 is unchanged whether P_1 is running σ_1' or ρ_1. Furthermore, the output of P_1 when running σ_1' is equal to the correct secret. Thus, the utility of P_1 when running σ_1' is at least the utility of P_1 when running ρ_1.
- If P_2 aborts first, the outputs of both P_1 and P_2 are identical whether P_1 runs σ_1' or ρ_1; this follows because as soon as P_2 aborts, strategy σ_1' "switches" to playing strategy ρ_1.

So, the utility obtained by playing σ_1' can only possibly be less than the utility obtained by playing ρ_1 when abort occurs. The maximum difference in the utilities in this case is $U^+ - U^-$. ■

The next claim shows that abort occurs at least as often when ρ_1 interacts with σ_2 as when ρ_1 interacts with $\hat{\rho}_2$.

Claim 3 $p_{\text{abort}}(\sigma_2) \geq p_{\text{abort}}(\hat{\rho}_2)$.

Proof Consider some view of ρ_1 on which it aborts first when interacting with $\hat{\rho}_2$. (The view includes both the information d_1 obtained from the dealer as well as the messages from P_2.) Since ρ_1 aborts first and, in every iteration, there is a *unique* non-aborting message that P_2 can send, it follows that ρ_1 will also abort first when interacting with σ_2 (who never aborts first) whenever ρ_1 is given d_1 from the dealer. The claim follows. ■

Define $U^* \stackrel{\text{def}}{=} \beta \cdot U^+ + (1-\beta) \cdot U_{\text{random}}$ and recall that $U^* < U$ by assumption. Now,

$$\begin{aligned} U_1(\rho_1, \rho_2) &= (1-\delta) \cdot U_1(\rho_1, \sigma_2) + \delta \cdot U_1(\rho_1, \hat{\rho}_2) \\ &\leq (1-\delta) \cdot \big(U + (U^* - U) \cdot p_{\text{abort}}(\sigma_2)\big) + \delta \cdot U_1(\rho_1, \hat{\rho}_2) + \mathsf{negl}(k), \end{aligned}$$

using Equation (3) and Claim 1. Also,

$$\begin{aligned} U_1(\sigma_1', \rho_2) &= (1-\delta) \cdot U_1(\sigma_1', \sigma_2) + \delta \cdot U_1(\sigma_1', \hat{\rho}_2) \\ &= (1-\delta) \cdot U + \delta \cdot U_1(\sigma_1', \hat{\rho}_2)\,. \end{aligned}$$

It follows that

$$\begin{aligned} & U_1(\rho_1, \rho_2) - U_1(\sigma_1', \rho_2) \\ &\quad = (1-\delta) \cdot (U^* - U) \cdot p_{\text{abort}}(\sigma_2) + \delta \cdot \big(U_1(\rho_1, \hat{\rho}_2) - U_1(\sigma_1', \hat{\rho}_2)\big) + \mathsf{negl}(k) \\ &\quad \leq (1-\delta) \cdot (U^* - U) \cdot p_{\text{abort}}(\hat{\rho}_2) + \delta \cdot (U^+ - U^-) \cdot p_{\text{abort}}(\hat{\rho}_2) + \mathsf{negl}(k)\,, \end{aligned}$$

using Claims 2 and 3. Since $U^* - U$ is strictly negative, there exists $\delta > 0$ for which the above expression is negligible for k large enough. This completes the proof.

B Game-Theoretic Definitions for the Multi-Party Setting

Throughout this section, Π is a t-out-of-n secret-sharing scheme and σ_i denotes the prescribed actions of P_i in the reconstruction phase. We begin by giving definitions for the case of single-player deviations, and then consider the case of coalitions.

B.1 Single-Player Deviations

Following standard notation, we let $\vec{\sigma} = (\sigma_{i_1}, \ldots, \sigma_{i_\ell})$ denote a vector of strategies with the indices $i_1, \ldots, i_\ell$ in some (implicit) index set I. For $\vec{\sigma}$ of this sort and $i^* \in I$, we set $\vec{\sigma}_{-i^*} = (\sigma_i)_{i \in I \setminus \{i^*\}}$.

Definition 6 Π induces a *computational Nash equilibrium* if for any set $I = \{i_1, \ldots, i_{t^*}\}$ of $t^* \geq t$ active parties, any $i \in I$, and any PPT strategy σ_i' we have that $U_i(\sigma_i', \vec{\sigma}_{-i}) \leq U_i(\vec{\sigma}) + \mathsf{negl}(k)$. ◇

We next define the notion of two strategies yielding equivalent play with respect to Π:

Definition 7 Fix $I \subseteq [n]$, an index $i \in I$, and a strategy ρ_i. Define view_{-i}^{Π} as follows:

> All parties play their prescribed strategies. Let trans denote the messages sent by P_i *not including any messages sent by P_i after it writes to its output tape.* Then view_{-i}^{Π} includes the information given by the dealer to all parties in $I \setminus \{i\}$, the random coins of all parties in $I \setminus \{i\}$, and the (partial) transcript trans.

Given algorithm T, define the random variable $\mathsf{view}_{-i}^{T,\rho_i}$ as follows:

> P_i and the parties in $I \setminus \{i\}$ interact, with P_i playing ρ_i and the other parties following their prescribed strategies. Let trans denote the messages sent by P_i. Algorithm T, given the entire view of P_i, outputs an arbitrary *truncation* trans' of trans. Then $\mathsf{view}_{-i}^{T,\rho_i}$ includes the information given by the dealer to all parties in $I \setminus \{i\}$, the random coins of all parties in $I \setminus \{i\}$, and the (partial) transcript trans'.

Strategy ρ_i *yields equivalent play with respect to* Π, denoted $\rho_i \approx \Pi$, if there exists a PPT algorithm T such that for all PPT distinguishers D

$$\left|\Pr[D(1^k, \mathsf{view}_{-i}^{T,\rho_i}) = 1] - \Pr[D(1^k, \mathsf{view}_{-i}^{\Pi}) = 1]\right| \leq \mathsf{negl}(k). \qquad \diamondsuit$$

Definition 8 Π induces a *computational strict Nash equilibrium* if

1. Π induces a computational Nash equilibrium;
2. For any $I \subseteq [n]$ with $|I| \geq t$, any $i \in I$, and any PPT strategy σ'_i for which $\sigma'_i \not\approx \Pi$, there is a $c > 0$ such that $U_i(\vec{\sigma}) \geq U_i(\sigma'_i, \vec{\sigma}_{-i}) + 1/k^c$ for infinitely many values of k. $\diamondsuit$

We next turn to defining stability with respect to trembles. We say that $\vec{\rho}_{-i}$ *is* δ*-close to* $\vec{\sigma}_{-i}$ if $\vec{\rho}_{-i}$ takes the following form: with probability $1-\delta$ all parties play according to $\vec{\sigma}_{-i}$, while with probability δ all parties follow an arbitrary (possibly correlated) PPT strategy σ'_{-i}. In this case, we refer to $\vec{\sigma}'_{-i}$ as the *residual strategy* of $\vec{\rho}_{-i}$.

Definition 9 Π induces a *computational Nash equilibrium that is stable with respect to trembles* if

1. Π induces a computational Nash equilibrium;
2. There is a noticeable function δ such that for any $I \subseteq [n]$ with $|I| \geq t$, any $i \in I$, any vector of PPT strategies $\vec{\rho}_{-i}$ that is δ-close to $\vec{\sigma}_{-i}$, and any PPT strategy ρ_i, there exists a PPT strategy $\sigma'_i \approx \Pi$ such that $U_i(\rho_i, \vec{\rho}_{-i}) \leq U_i(\sigma'_i, \vec{\rho}_{-i}) + \mathsf{negl}(k)$. $\diamondsuit$

B.2 Coalitions

We view a coalition $\mathcal{C}$ as a set of parties who may coordinate their strategies in an arbitrary way. Since the coalition acts in unison, we treat the utility of the coalition as a whole and, in particular, view the coalition as having only a single output value (rather than viewing each member of the coalition as potentially outputting a different value). Let $\mu_{\mathcal{C}}(\cdot)$ denote the utility of the coalition $\mathcal{C}$. As before, we assume the following utilities:

1. If o is an outcome in which $\mathcal{C}$ learns the secret and no player outside $\mathcal{C}$ does, then $\mu_{\mathcal{C}}(o) = U^+$.
2. If o is an outcome in which all parties active during the reconstruction phase (including $\mathcal{C}$) learn the secret, then $\mu_{\mathcal{C}}(o) = U$.
3. If o is an outcome in which $\mathcal{C}$ does not learn the secret, then $\mu_{\mathcal{C}}(o) = U^-$.

If $\vec{\sigma} = (\sigma_{\mathcal{C}}, \vec{\sigma}_{-\mathcal{C}})$ then $U_{\mathcal{C}}(\vec{\sigma})$ denotes the expected utility of $\mathcal{C}$ when parties in $\mathcal{C}$ follow $\sigma_{\mathcal{C}}$ and the remaining parties follow $\sigma_{-\mathcal{C}}$.

Definition 10 Π induces an r*-resilient computational Nash equilibrium* if for any $I \subseteq [n]$ with $|I| \geq t$, any $\mathcal{C} \subset I$ with $|\mathcal{C}| \leq r$, and any PPT strategy $\sigma'_{\mathcal{C}}$ we have $U_{\mathcal{C}}(\sigma'_{\mathcal{C}}, \vec{\sigma}_{-\mathcal{C}}) \leq U_{\mathcal{C}}(\vec{\sigma}) + \mathsf{negl}(k)$. $\diamondsuit$

We define the notion of two coalition strategies $\sigma_{\mathcal{C}}, \sigma'_{\mathcal{C}}$ yielding equivalent play in a manner analogous to Definition 7, except that now the transcript included in $\mathsf{view}_{-\mathcal{C}}^{\Pi}$ does not include messages sent by the parties in $\mathcal{C}$ once *any* party in $\mathcal{C}$ writes its output.

Definition 11 Π induces an *r-resilient computational strict Nash equilibrium* if

1. Π induces an r-resilient computational Nash equilibrium;
2. For any $\mathcal{C} \subset I \subseteq [n]$ with $|I| \geq t$ and $|\mathcal{C}| \leq r$, and any PPT strategy $\sigma'_{\mathcal{C}}$ for which $\sigma'_{\mathcal{C}} \not\approx \Pi$, there is a $c > 0$ such that $U_{\mathcal{C}}(\vec{\sigma}) \geq U_{\mathcal{C}}(\sigma'_{\mathcal{C}}, \vec{\sigma}_{-\mathcal{C}}) + 1/k^c$ for infinitely many values of k. $\diamondsuit$

Definition 12 Π induces an *r-resilient computational Nash equilibrium that is stable with respect to trembles* if

1. Π induces an r-resilient computational Nash equilibrium;

2. There is a noticeable function δ such that for any $I \subseteq [n]$ with $|I| \geq t$, any $\mathcal{C} \subset I$ with $|\mathcal{C}| \leq r$, any vector of PPT strategies $\vec{\rho}_{-\mathcal{C}}$ that is δ-close to $\vec{\sigma}_{-C}$, and any PPT strategy $\rho_{\mathcal{C}}$, there exists a PPT strategy $\sigma'_{\mathcal{C}} \approx \Pi$ such that $U_{\mathcal{C}}(\rho_{\mathcal{C}}, \vec{\rho}_{-\mathcal{C}}) \leq U_{\mathcal{C}}(\sigma'_{\mathcal{C}}, \vec{\rho}_{-\mathcal{C}}) + \mathsf{negl}(k)$. $\diamondsuit$

C The Asynchronous Case

We begin with a few technical notes as to how we model the asynchronous setting:

1. As is standard in the asynchronous setting, we allow messages to be delayed and to be delivered in arbitrary order, but we assume eventual message delivery. (I.e., a message sent from one party to another will be received by time $t = \infty$.)

2. We define an outcome o by the values recorded on the output tapes of the players at time $t = \infty$. In particular, it does not matter whether a party halts in a finite number of steps or not; all that matters is the value that is eventually written on its output tape (see the next item). This is essential, as even parties who follow the protocol honestly are not guaranteed to output the correct secret in any fixed time bound, and a party who deviates from the protocol can cause an honest party to run indefinitely.

3. Due to the above, we allow parties to write a value to their output tape multiple times. We stress, however, that the value that "counts" as far as defining the outcome o is the value on a party's output tape at time $t = \infty$.

4. We allow cheating players to schedule message delivery in the network. The definition of "yielding equivalent play" (cf. Definition 2), however, still refers only to the actual messages sent by the players, and not to the way message delivery is scheduled. (We cannot hope to claim that changing the order of message delivery decreases a party's utility.)

We also modify the definition of "yielding equivalent play" to account for the asynchronous communication. A message sent by P_j is recursively defined to *affect* P_i if either (1) it is sent to P_i, or (2) it is sent to a party P_k who subsequently send a message that affects P_i. Our definition of "yielding equivalent play" (in the case of single-player deviations) is as follows:

Definition 13 Fix $I \subseteq [n]$, an index $i \in I$, and a strategy ρ_i. Define view^{Π}_{-i} as follows:

> All parties play their prescribed strategies. Let trans denote the messages sent by P_i *not including those messages that do not affect* P_i. Then view^{Π}_{-i} includes the information given by the dealer to all parties in $I \setminus \{i\}$, the random coins of all parties in $I \setminus \{i\}$, and the (partial) transcript trans.

Given algorithm T, the random variable $\mathsf{view}_{-i}^{T,\rho_i}$ is defined as in Definition 7. Strategy ρ_i *yields equivalent play with respect to* Π, denoted $\rho_i \approx \Pi$, if there exists a PPT algorithm T such that for all PPT distinguishers D

$$\left|\Pr[D(1^k, \mathsf{view}_{-i}^{T,\rho_i}) = 1] - \Pr[D(1^k, \mathsf{view}_{-i}^{\Pi}) = 1]\right| \leq \mathsf{negl}(k). \qquad \diamondsuit$$

Sharing Phase

The sharing phase is identical to protocol $\Pi_{t,n}$ in Figure 2.

Reconstruction Phase

Let I denote the set of the indices of the t active players. Each party P_i (for $i \in I$) chooses $s_i^{(0)}$ uniformly from $\{0,1\}^\ell$ and writes it on its output tape. For $r = 1, \ldots,$ party P_i does:

- P_i sends the following to all players:

$$\left(y_i^{(r)} := \mathsf{Eval}_{sk_i}(r),\ z_i^{(r)} := \mathsf{Eval}'_{sk'_i}(r),\ \mathsf{Prove}_{sk_i}(r),\ \mathsf{Prove}'_{sk'_i}(r) \right).$$

- If P_i receives an incorrect proof from some other party P_j, then P_i terminates. (Note that if this occurs then the value $s_i^{(r-1)}$ is already written on its output tape.) Otherwise, as soon as P_i receives $t-1$ valid messages for iteration r it does:
  - P_i sets $h_j^{(r)} := h_j \oplus z_j^{(r)}$ for all $j \in I$, and interpolates a degree-$(t-1)$ polynomial $H^{(r)}$ through the t points $\{h_j^{(r)}\}_{j \in I}$. If $H^{(r)}(0) = 0$ then P_i terminates after sending its current-iteration message. (Note that if this occurs then the value $s_i^{(r-1)}$ is already written on its output tape.)
  - Otherwise, P_i sets $g_j^{(r)} := g_j \oplus y_j^{(r)}$ for all $j \in I$, and interpolates a degree-$(t-1)$ polynomial $G^{(r)}$ through the points $\{g_j^{(r)}\}_{j \in I}$. It writes the value $s_i^{(r)} := G^{(r)}(0)$ to its output tape and proceeds to the next iteration.

Figure 3: Protocol $\Pi'_{t,n}$ for "exactly t-out-of-n" secret sharing in the asynchronous case.

Protocol $\Pi'_{t,n}$ for the asynchronous case is presented in Figure 3. We now sketch the proof that this protocol induces a $(t-1)$-resilient computational Nash equilibrium whenever exactly t parties are active in the reconstruction phase. (Once again, the fact that there is always a unique legal message furthermore implies that the protocol induces a $(t-1)$-resilient computational *strict* Nash equilibrium that is stable with respect to trembles.) Assume some set of t parties I running the reconstruction phase, and consider some coalition $\mathcal{C} \subset I$ of size at most $t-1$. Let P^* be any player in I who is not in $\mathcal{C}$. As usual, the best strategy for $\mathcal{C}$ is to not abort until it can definitively identify iteration r^*, which occurs only after it receives the iteration-(r^*+1) message from P^*. But P^* only sends its iteration-(r^*+1) message after it has received (valid) iteration-r^* messages from all the parties in $\mathcal{C}$. By this point, no matter what the parties in $\mathcal{C}$ do, P^* has the correct secret s written on its output tape.