Efficient Rational Secret Sharing in Standard Communication Networks

Georg Fuchsbauer[1,⋆], Jonathan Katz[2,⋆⋆], and David Naccache[1]

[1] École Normale Supérieure, LIENS - CNRS - INRIA, Paris, France
{georg.fuchsbauer,david.naccache}@ens.fr
[2] University of Maryland, USA
jkatz@cs.umd.edu

Abstract. We propose a new methodology for rational secret sharing leading to various instantiations (in both the two-party and multi-party settings) 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.

We also propose new equilibrium notions (namely, computational versions of *strict Nash equilibrium* and *stability with respect to trembles*) and prove that our protocols satisfy them. These notions guarantee, roughly speaking, that at each point in the protocol there is a *unique* legal message a party can send. This, in turn, ensures that protocol messages cannot be used as subliminal channels, something achieved in prior work only by making strong assumptions on the communication network.

1 Introduction

The classical problem of *t-out-of-n secret sharing* [28,5] involves a dealer D who distributes shares of a secret s to players $P_1, \ldots, P_n$ so that (1) any t or more players can reconstruct the secret without further involvement of the dealer, yet (2) any group of fewer than t players gets no information about s. For example, in Shamir's scheme [28] 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, t players broadcast their shares and interpolate the polynomial. 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

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-11799-2_36

⋆ Work supported by EADS, the French ANR-07-SESU-008-01 PAMPA Project, and the European Commission through the ICT Program under Contract ICT-2007-216646 ECRYPT II.

⋆⋆ Work done while visiting École Normale Supérieure 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.

D. Micciancio (Ed.): TCC 2010, LNCS 5978, pp. 419–436, 2010.

when reconstruction of the secret is desired. Beginning with the work of Halpern and Teague [13], protocols for secret sharing and other cryptographic tasks have begun to be re-evaluated in a game-theoretic light (see [7,16] 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 [13] 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 is supposed to broadcast their share simultaneously. It is easy to see, however, that each player does no worse (and potentially does better) by withholding their share no matter what the other players do. Consider P_1: If fewer than $t-1$ players reveal their shares, 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* 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.

Several works [13,11,24,1,18,19,27,26,4] have 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. [20,21, 15,14]. Each of these works has some or all of the following disadvantages:

On-line dealer or trusted/honest parties. Halpern and Teague [13] 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 Halpern-Teague solution also applies only when $t, n \geq 3$.) Ong et al. [27] assume that sufficiently many parties behave honestly during the reconstruction phase.

Computational inefficiency. To eliminate the on-line dealer, several schemes rely on multiple invocations of protocols for generic secure multi-party computation [11,24,1,18,4]. It is unclear whether computationally *efficient* protocols with suitable functionality can be designed. The solutions of [20,21,15,14], though following a different high-level approach, also rely on generic secure multi-party computation.

Strong communication models. All prior schemes for $n > 2$ assume broadcast. The solutions in [13,11,24,1] 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 [20,21, 15,14] rely on *physical* assumptions such as secure envelopes and ballot boxes.

Kol and Naor [18] 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; this approach cannot (efficiently) handle secrets of super-logarithmic length. Subsequent work by Kol and Naor [19] (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 [19] in Section 1.3.

1.1 Our Results

We show protocols for both 2-out-of-2 and t-out-of-n secret sharing (resilient to coalitions of size $t-1$) that 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.

The primary advantage of our protocols, however, is that they do not require broadcast or simultaneous communication but can instead rely on synchronous (but *non*-simultaneous) point-to-point channels. Recall that all prior schemes for $n > 2$ assume broadcast; furthermore, 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. Going further, we show that our protocol can be adapted for *asynchronous* point-to-point networks (with respect to a natural extension of the model for rational secret sharing), thus answering a question that had been open since the work of Halpern and Teague [13].

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.) The first notion we introduce is a computational version of *strict* Nash equilibrium. A similar notion was put forth by Kol and Naor [19], 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; doing so allows us to circumvent the limitations that hold in the information-theoretic case. We also formalize a notion of *stability with respect to trembles*, motivated by [16]; a different formalization of this notion, with somewhat different motivation, is given in [27].

Our definitions effectively rule out "signalling" via subliminal channels in the protocol. In fact, our protocols ensure that, at every point, *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., [15,21,3,2]), which achieved it only by relying on physical assumptions [15,21] or non-standard network models [3,2].

1.2 Overview of Our Approach

We follow the same high-level approach as in [13,11,24,1,18,19,4]. Our reconstruction protocol proceeds in a sequence of "fake" iterations followed by a single "real" iteration. Roughly speaking:

- 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 [11,24,1,18], a secure multi-party computation is 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) [25], 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 Appendix A for definitions of VRFs.) 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 [11,24,1,18] allows parties to identify the real iteration *as soon as it occurs*. We could use this approach for our protocol as well if we assumed simultaneous channels, 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.

Motivated by recent work on fairness (in the malicious setting) [10,12], 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.

Our protocol assumes parties have no auxiliary information about the secret s. (If simultaneous channels are assumed, then our protocol *does* tolerate auxiliary information about s.) We believe there are settings where this assumption is valid, and that understanding this case sheds light on the general question of rational computation. Prior work in the non-simultaneous model [18, 19] also fails in the presence of auxiliary information, and in fact this is inherent [4].

1.3 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 [19] (an extension of this protocol is given in [4]). 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 [18].) Moreover, the *expected* length of the parties' shares in their 2-out-of-2 scheme is $O(\beta^{-1} \cdot (|s| + k))$ (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 [19, 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. A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible* if for all $c > 0$ there is a $k_c > 0$ such that $\epsilon(k) < 1/k^c$ for all $k > k_c$; 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 $k > k_c$.

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 an information-theoretic sense.

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 consider 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. Our notion is also 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 [13] 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 for the outcome o, described from the point of view of P_i (we could also work with utilities satisfying the more general constraints above, as long as utilities are known):

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

Our conditions impose $U^+ > U > U^-$. Define

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

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.

In contrast to [4], we make no distinction between outputting the wrong secret and outputting a special "don't know" symbol; both are considered a failure to output the correct secret. By adapting techniques from their work, however, we can incorporate this distinction as well (as long as the relevant utilities are known). See Remark 2 in Section 3.

Strategies in our context refer to probabilistic polynomial-time interactive Turing machines. Given a vector of strategies $\boldsymbol{\sigma}$ for t^* parties active in the reconstruction phase, let $u_i(\boldsymbol{\sigma})$ denote the *expected* utility of P_i. (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 the standard game-theoretic notation, we define $\boldsymbol{\sigma}_{-i} \stackrel{\text{def}}{=} (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_{t^*})$ and $(\sigma_i', \boldsymbol{\sigma}_{-i}) \stackrel{\text{def}}{=} (\sigma_1, \ldots, \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots, \sigma_{t^*})$; that is, $(\sigma_i', \boldsymbol{\sigma}_{-i})$ denotes the strategy vector $\boldsymbol{\sigma}$ 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 [13], 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 [11,24,1].

The notion of surviving iterated deletion, though, is also problematic in several respects. Kol and Naor [19] show a secret-sharing protocol that is "intuitively bad" yet technically satisfies the definition because no strategy weakly dominates any other. Also, a notion of surviving iterated deletion taking *computational* issues into account has not yet been defined (and doing so appears difficult). See [16,17] for other arguments against this notion.

Motivated by these drawbacks (and more), researchers have proposed other strengthenings of the Nash equilibrium concept [16,18,19]. Kol and Naor define *resistance to backward induction* [18], *everlasting equilibrium*, and *strict Nash equilibrium* [19]. The latter two notions are defined information-theoretically, and are overly conservative in that they rule out some "natural" protocols using cryptography. 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 for 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: since *any* (detectable) deviation from the protocol results in lower utility (when other parties follow the protocol), a party who tries to use protocol messages as a covert channel risks losing utility if there is any reasonable probability that other players are following the protocol.

We propose here a *computational* version of strict Nash equilibrium. Our definition retains the intuitive appeal of strict Nash, while meaningfully taking computational limitations into account. Moreover, our protocols satisfy the following, stronger condition: at every point in the protocol, there is a *unique* legal message that each party can send. Our protocols thus rule out subliminal communication in a strong sense, an explicit goal in prior work [20,22,21,3].

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 [16], but we flesh out the (non-trivial) technical details. Another formulation (*trembling-hand perfect equilibrium*), with somewhat different motivation, is discussed in [27].

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

We focus on the two-party case; the multi-party case is treated in the full version of this work [9]. Here, Π is a 2-out-of-2 secret-sharing scheme and σ_i is the prescribed action of P_i in the reconstruction phase.

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.*

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 but not including any messages sent by P_1 after it writes to its (write-once) 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*

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

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 its 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).*

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 "most of the time," 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 (undetectable) *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. The fact that the prescribed strategies are in Nash equilibrium ensures that 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).*

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).

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

Let $\mathcal{S} = \{0,1\}^\ell$ be the domain of the secret. 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 first computes the keys $(pk_1, sk_1), (pk_2, sk_2) \leftarrow \mathsf{Gen}(1^k)$ as well as $(pk_1', sk_1'), (pk_2', sk_2') \leftarrow \mathsf{Gen}'(1^k)$. Then the dealer computes:

[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 is a strategy σ_i' that *does* yield equivalent play w.r.t. Π and performs about as well.

– $\mathsf{share}_1 := \mathsf{Eval}_{sk_2}(i^*) \oplus s$ 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 $(sk_2, sk'_2, pk_1, pk'_1, \mathsf{share}_2, \mathsf{signal}_2)$.

As written, the share given to each party only hides s in a *computational* sense. Nevertheless, information-theoretic secrecy is easy to achieve; see Remark 1 at the end of this section.

Reconstruction phase

At the outset, 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

– $y_2^{(i)} := \mathsf{Eval}_{sk_2}(i)$, $\pi_2^{(i)} := \mathsf{Prove}_{sk_2}(i)$
– $z_2^{(i)} := \mathsf{Eval}'_{sk'_2}(i)$, $\bar{\pi}_2^{(i)} := \mathsf{Prove}'_{sk'_2}(i)$.

It then 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 $s_1^{(i-1)}$, sends 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

– $y_1^{(i)} := \mathsf{Eval}_{sk_1}(i)$, $\pi_1^{(i)} := \mathsf{Prove}_{sk_1}(i)$
– $z_1^{(i)} := \mathsf{Eval}'_{sk'_1}(i)$, $\bar{\pi}_1^{(i)} := \mathsf{Prove}'_{sk'_1}(i)$.

It then 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.

Fig. 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. Then 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 expected utility of this strategy is strictly less than U, in which case P_1 has no incentive to deviate.

That Π induces a *strict* computational Nash equilibrium (which is also stable with respect to trembles) follows since there is always a *unique* valid message a party can send; anything else is treated as an abort. A proof of the following theorem appears in the full version of this work [9].

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

Remark 1. The sharing phase, as described, guarantees computational secrecy only. A generic transformation from any such protocol (with bounded-size shares) to one that achieves information-theoretic secrecy follows: After D generates shares s_1, s_2 in the computationally secure scheme, it chooses random r_1, r_2 and random keys k_1, k_2, and gives to P_i the share $(s_i \oplus r_i,\ k_i,\ r_{3-i},\ \mathsf{MAC}_{k_{3-i}}(r_{3-i}))$, where MAC denotes an information-theoretically secure MAC. The reconstruction phase begins by having the parties exchange r_1 and r_2 along with the associated MAC tags, verifying the tags (and aborting if they are incorrect), and then recovering s_1, s_2. They then run the original protocol. It is easy to see that this maintains all the game-theoretic properties of the original protocol.

Remark 2. In the reconstruction phase, as described, one party can cause the other to output an incorrect secret (by aborting early). If the utilities of doing

so are known, the protocol can be modified to rule out this behavior (in a game-theoretic sense) using the same techniques as in [4, Section 5.2]. Specifically, the dealer can — for each party — designate certain rounds as "completely fake", so that the party will know to output $\perp$ instead of an incorrect secret in case the other party aborts in that round. Using VRFs, this still can be achieved with bounded-size shares. Details will appear in the full version.

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 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 the full version of this work [9].

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.

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] VRF keys $(pk_1, sk_1), \ldots, (pk_n, sk_n) \leftarrow \mathsf{Gen}(1^k)$ followed by $(pk'_1, sk'_1), \ldots, (pk'_n, sk'_n) \leftarrow \mathsf{Gen}'(1^k)$.
- 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. $\{(pk_j, pk'_j)\}_{1 \le j \le n}$
  2. $\{g_j := G(j) \oplus \mathsf{Eval}_{sk_j}(r^*)\}_{1 \le j \le n}$
  3. $\{h_j := H(j) \oplus \mathsf{Eval}'_{sk'_j}(r^*+1)\}_{1 \le j \le n}$

Reconstruction Phase

Let I be the 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 computes $s_i^{(r)}$ 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 $\{0,1\}^k$.

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

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. *If $\beta > 0$ and $U > \beta \cdot U^+ + (1-\beta) \cdot U_{\text{random}}$, 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 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])

Asynchronous networks. Our protocol $\Pi_{t,n}$ can be adapted to work even when the parties communicate over an *asynchronous* point-to-point network. (In our model of asynchronous networks, messages can be delayed arbitrarily and delivered out of order, but any message that is sent is eventually delivered.) In this 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 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 given in the full version of this work [9].

As before, we can handle the general case by having the dealer run the "exactly t^*-out-of-n" protocol just described for all values of $t^* \in \{t, \ldots, n\}$.

[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 when t (or $n-t$) is small.

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: 25th ACM Symposium Annual on Principles of Distributed Computing, pp. 53–62. ACM Press, New York (2006)
2. Alwen, J., Katz, J., Lindell, Y., Persiano, G., Shelat, A., Visconti, I.: Collusion-free multiparty computation in the mediated model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 524–540. Springer, Heidelberg (2009)
3. Alwen, J., Shelat, A., Visconti, I.: Collusion-free protocols in the mediated model. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 497–514. Springer, Heidelberg (2008)
4. Asharov, G., Lindell, Y.: Utility dependence in correct and fair rational secret sharing. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 559–576. Springer, Heidelberg (2009), A full version, containing additional results: http://eprint.iacr.org/209/373
5. Blakley, G.: Safeguarding cryptographic keys. In: Proc. AFIPS National Computer Conference, vol. 48, pp. 313–317 (1979)
6. Dodis, Y.: Efficient construction of (distributed) verifiable random functions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 1–17. Springer, Heidelberg (2002)
7. Dodis, Y., Rabin, T.: Cryptography and game theory. In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.) Algorithmic Game Theory, pp. 181–207. Cambridge University Press, Cambridge (2007)
8. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
9. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient rational secret sharing in standard communication networks, http://eprint.iacr.org/2008/488
10. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 413–422. ACM Press, New York (2008)
11. Gordon, S.D., Katz, J.: Rational secret sharing, revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)
12. Gordon, S.D., Katz, J.: Partial fairness in secure two-party computation (2008), http://eprint.iacr.org/2008/206
13. Halpern, J., Teague, V.: Rational secret sharing and multiparty computation: Extended abstract. In: 36th Annual ACM Symposium on Theory of Computing (STOC), pp. 623–632. ACM Press, New York (2004)
14. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably secure devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)
15. Izmalkov, S., Micali, S., Lepinski, M.: Rational secure computation and ideal mechanism design. In: 46th Annual Symposium on Foundations of Computer Science (FOCS), pp. 585–595. IEEE, Los Alamitos (2005)
16. Katz, J.: Bridging game theory and cryptography: Recent results and future directions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 251–272. Springer, Heidelberg (2008)
17. Katz, J.: Ruminations on defining rational MPC. Talk given at SSoRC, Bertinoro, Italy (2008), http://www.daimi.au.dk/~jbn/SSoRC2008/program

18. Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg (2008)
19. Kol, G., Naor, M.: Games for exchanging information. In: 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 423–432. ACM Press, New York (2008)
20. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely fair SFE and coalition-safe cheap talk. In: 23rd ACM Symposium Annual on Principles of Dis- tributed Computing, pp. 1–10. ACM Press, New York (2004)
21. Lepinski, M., Micali, S., Shelat, A.: Collusion-free protocols. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 543–552. ACM Press, New York (2005)
22. Lepinski, M., Micali, S., Shelat, A.: Fair-zero knowledge. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 245–263. Springer, Heidelberg (2005)
23. Lysyanskaya, A.: Unique signatures and verifiable random functions from the DH-DDH separation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 597–612. Springer, Heidelberg (2002)
24. Lysyanskaya, A., Triandopoulos, N.: Rationality and adversarial behavior in multi-party computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 180–197. Springer, Heidelberg (2006)
25. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th Annual Symposium on Foundations of Computer Science (FOCS), pp. 120–130. IEEE, Los Alamitos (1999)
26. Miclai, S., Shelat, A.: Truly rational secret sharing. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 54–71. Springer, Heidelberg (2009)
27. Ong, S.J., Parkes, D., Rosen, A., Vadhan, S.: Fairness with an honest minority and a rational majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)
28. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)

A 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 [25], and various constructions in the standard model are known [25, 6, 23, 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; see Section 3.1.)

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* $x \notin \{x_1, \ldots, x_\ell\}$*.*
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' *and* $\mathsf{succeeds}$ *if* $b' = b$*.*

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