Scalable Mechanisms for Rational Secret Sharing

Varsha Dani * Yamel Rodriguez * Jared Saia *

Abstract

We consider the classical secret sharing problem in the case where all agents are selfish but rational. In recent work, Kol and Naor show that in the non-simultaneous communciation model (i.e. when rushing is possible), there is no Nash equilibrium that ensures all agents learn the secret. However, they describe a mechanism for this problem that is an ϵ-*Nash equilibrium*, i.e. it is close to an equilibrium in the sense that no player can gain more than ϵ utility by deviating from it.

Unfortunately, the Kol and Naor mechanism, and, to the best of our knowledge, all previous mechanisms for this problem require each agent to send $O(n)$ messages in expectation, where n is the number of agents. This may be problematic for some applications of rational secret sharing such as secure multiparty computation and simulation of a mediator.

We address this issue by describing a mechanism for rational n-out-of-n secret sharing that is an ϵ-Nash equilibrium, and is *scalable* in the sense that it requires each agent to send only an expected $O(1)$ messages and polylogarithmic number of bits. Moreover, the latency of our mechanism is $O(\log n)$ in expectation, compared to $O(n)$ expected latency for the Kol and Naor result. We also design scalable mechanisms for a relaxed variant of rational m-out-of-n secret sharing where $m = \theta(n)$. Our mechanisms are non-cryptographic, and are not susceptible to backwards induction.

"Three can keep a secret if two are dead." - Benjamin Franklin

PODC regular submission with consideration as a brief announcement otherwise. This paper should **not** be considered for the best student paper award.

*Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386; email: {varsha, yamel, saia}@cs.unm.edu. This research was partially supported by NSF CAREER Award 0644058, NSF CCR-0313160, and an AFOSR MURI grant.

1 Introduction

Secret sharing is one of the most fundamental problems in security, and is an important primitive in many cryptographic protocols, including secure multiparty computation. Recently, there has been interest in solving *rational secret sharing* [8, 4, 5, 1, 11]. In this setting, there are n selfish but rational agents, and we want to distribute shares of a secret to each agent, and design a protocol for the agents ensures that: 1) if any group of m agents follow the protocol they will all learn the secret; and 2) knowledge of less than m of the shares reveals nothing about the secret. Moreover, we want our protocol to be a *Nash equilibrium* in the sense that no player can improve their utility by deviating from the protocol, given that all other players are following the protocol.

Unfortunately, all previous solutions to this problem require each agent to send $O(n)$ messages in expectation, and so do not scale to large networks. Rational secret sharing is a primitive for rational multiparty computation, which can be used to compute an arbitrary function in a completely decentralized manner, without a trusted external party. A typical application of rational multiparty computation might be to either run an auction, or to hold a lottery to assign resources in a network. It is easy to imagine such applications where the number of players is large, and where it is important to have algorithms whose bandwidth and latency costs scale well with the number of players. Moreover, in a game theoretic setting, standard tricks to circumvent scalability issues, like running the protocol only on a small subset of the players, may be undesirable since they could lead to increased likelihood of bribery attacks.

In this paper, we address this issue by designing scalable mechanisms for rational secret sharing. Our main result is a protocol for rational n-out-of-n secret sharing that 1) requires each agent to send only an expected $O(1)$ messages and polylogarithmic bits; and 2) has $O(\log n)$ expected latency. We also design scalable mechanisms for a relaxed variant of m-out-of-n rational secret sharing in the case where m is $\theta(n)$.

1.1 The Problem

We assume there are n rational but selfish player. The players' utility functions are such that they prefer to learn the secret, but also prefer that other players not learn the secret. Following previous work [8, 4, 5, 11], we assume all the players have the same utility function, which is specified by three constants U_+, U, and U_-. Here U_+ is the utility to a player if he alone learns the secret, U is the utility if he learns the secret but at least one other player learns it as well, and finally U_- is the utility when the player does not learn the secret. We further assume that $U_+ > U > U_-$, so that the players' preferences are strict. We note that, as in previous work, the utility function does not distinguish the cases where different numbers of other players may learn the secret.

We will assume that the secret is an arbitrary element of a large (fixed) finite set $\mathcal{S}$. The shares are provided to the players by a dealer, who is active only once, at the beginning of the game. The players must then communicate with each other in order to reconstruct the secret.

We assume that the communication between the players is *point-to-point* and through private channels. In other words, if player A sends a message to player B, then a third player C is not privy to the message that was sent, or indeed even to the fact of a message having been sent. We assume communication is synchronous in that there is an upperbound known on the maximum amount of time required to send a message from one player to another. However, we assume *non-simultaneous* communication, and thus allow for the possibility of *rushing*, where a player may receive messages from other players in a round before sending out its own messages.

Our goal is to provide protocols for the dealer and rational players such that the players following

the protocol can reconstruct the secret. Moreover, we want a protocol that is *scalable* in the sense that the amount of communication and the latency of the protocol should be a slow growing function of the number of players.

We are also concerned with the problem of rational multiparty computation, which is an important application of n-out-of-n rational secret sharing. In the problem of rational multiparty computation, there are n rational agents, and each agent i has an input x_i to a function f of n variables. Each player prefers to learn the output of f, but also prefers that other players do not. In particular, the utility functions are the same as for the secret sharing problem. The goal is to design a protocol for rational players that will ensure that all players learn the output of f. As described in [5], the algorithm from [3] can be used to solve the problem of rational multiparty computation provided one has a solution to n-out-of-n rational secret sharing.

1.2 Our Results

In recent work, Kol and Naor show that in the non-simultaneous broadcast model (i.e. when rushing is possible), there is no Nash equilibrium that ensures all agents learn the secret [8]. They thus consider the case of designing an *ϵ-Nash equilibrium* for the problem in this communication model, where an ϵ-Nash equilibrium is close to an equilibrium in the sense that no player can gain more than ϵ utility by deviating from it. In addition, their protocol is Monte Carlo, succeeding with probability $1 - \delta$ for any fixed positive δ.

Our main result is a scalable, Monte Carlo protocol for n-out-of-n rational secret sharing that is also an ϵ-Nash equilibrium. This result is summarized in the following theorem.

Theorem 1.1. *For any fixed positive ϵ, δ there exists a protocol for rational n-out-of-n secret sharing that with probability $1 - \delta$ has the following properties:*

- *The protocol is an ϵ-Nash equilibrium*
- *All players learn the secret*
- *The protocol, in expectation, requires each player to send $O(1)$ messages and $O(\log |\mathcal{S}|)$ bits, and has latency $O(\log n)$*

As discussed above, the n-out-of-n case for rational secret sharing is a critical component of rational multiparty computation. Our result for n-out-of-n rational secret sharing enables a protocol for rational multiparty computation that is an ϵ-Nash equilibrium in the point-to-point, synchronous, non-simultaneous communication model. Moreover, it reduces worst case bandwidth by a multiplicative factor of n, and latency by a multiplicative factor of $\theta(n/\log n)$ over the rational multiparty protocol in this communication model from [8].

In this paper, we also consider the problem of m-out-of-n rational secret sharing for the case where $m < n$. Designing scalable algorithms for this problem is challenging because of the tension between reduced communication, and the need to ensure that *any* active set of m players can reconstruct the secret. For example, consider the case where each player sends $O(\log n)$ messages. If $m = o(n/\log n)$, even if the set of active players is chosen *randomly*, it is likely that there will be some active player that will never receive a message from any other active player. Moreover, even if $m = \theta(n)$, if the set of active players is chosen in a worst case manner, it is easy to see that a small subset of the active players can easily be isolated so that they never receive messages from the other active players, and are thus unable to reconstruct the secret.

Despite the difficulty of the problem, scalable rational secret sharing for the m-out-of-n case may still be of interest for applications like the Vanish peer-to-peer system [2]. To determine what might at least be possible, we consider a significantly relaxed variant of the problem. In particular, we require $m = \theta(n)$ and that the set of m active players be chosen independently of the random bits of the dealer. Our result is given in the following theorem, whose sketch is given in Section A.

Theorem 1.2. *For any fixed positive ϵ, δ, and $m = \theta(n)$, there exists a protocol for rational m-out-of-n secret sharing, which with probability $1 - \delta$ has the following properties, provided that the subset of active players is chosen independently of the random bits of the dealer:*

- *The protocol is a Nash equilibrium*
- *The protocol ensures that if at least a $m/n + \epsilon$ fraction of the players are active, all players will learn the secret; and if less than a $m/n - \epsilon$ fraction of the players are active, then the secret can not be recovered*
- *The protocol requires each player to send $O(\log n)$ messages, and $O(\log n \log |\mathcal{S}|)$ bits, and has latency $O(\log n)$*

1.3 Our Approach

Our protocol shares many similarities with the approach of Kol and Naor in [8]. Following are the techniques we use from their protocol. First, each player i receives a set of lists from the dealer, and the j-th element of each list is used in round j. Second, we make use of two lists from [8]: 1) the list L_i that contains potential secrets, for some value t^*, the t^*-th element in each of these lists is the true secret; and 2) the list S_i that contains shares of an indicator sequence, for each round j, that reveals whether or not the j-th round is the t^*-th round. Finally, we make use of a clever technique from [8] to avoid backwards induction. In particular, one player, chosen uniformly at random by the dealer, is designated the *short* player and the remaining players are designated the *long* players. The length of the short players lists is determined by geometric distributions, and the length of the long players' lists exceeds the length of the short player's list by an amount that is geometrically distributed. This scheme protects against backwards induction by making it difficult for any player to know exactly how many rounds the protocol will last.[1]

Following are the novel techniques in our approach. First, we restrict communication in our protocol to a certain type of complete binary tree; this is critical for ensuring that the total number of messages sent by each player is scalable. Second, we make use of an iterated secret sharing scheme over this tree to divide up shares of secrets among the players. This scheme is similar to that used in recent work by King and Saia [7] on the problem of scalable Byzantine agreement, and suggests deeper connections between the two problems. Finally, we make use of an iterated tag and hash scheme [13, 12] to enable checking of secrets and iterated shares of secrets. Tag and hash schemes were used previously in [8], but applying them in our setting, where communication is severely curtailed, is technically challenging.

1.4 Related Work

Since its introduction by Halpern and Teague in [5], there has been significant work on the problem of rational secret sharing, including results of Halpern and Teague [5], Gordon and Katz [4],

[1] It is also what makes their protocol and ours Monte Carlo, since with some small but constant probability, all the geometric random variables may be so small that it is better for each player to randomly try to guess the secret from the elements in their L_i list than to follow the protocol.

Abraham et al. [1], Lysyanskaya and Triandopoulos [11] and Kol and Naor [8]. All of this related work except for [8], assumes the existence of simultaneous communication, either by broadcast or private channels. Several of the protocols proposed [4, 1, 11] make use of cryptographic assumptions and achieve equilibria under the assumption that the players are computationally bounded. The protocol from [1] is robust to coalitions; and the protocol from [11] works in the situation where players may be either rational or adversarial.

The work of Kol and Naor [8] is closest to our own work in that they do not assume simultaneous communication, and do not make cryptographic assumptions. As we have already discussed, our protocols make use of several clever ideas from their result.

Additional work by Lepinski et al. [10, 9] and Izmalkov et al. [6] describe protocols for rational secure multiparty computation that are fair and robust to coalitions. However, their results rely on the physical assumption of "secure envelopes" [10], which seem difficult to implement for participants that are physically distant.

2 Our n-out-of-n Secret Sharing Protocol

We give an informal description of the dealer's and players' protocols in Sections 2.1 and 2.2. The formal descriptions of these protocols appears in Algorithm 1 and Algorithm 4.

2.1 Dealer's Protocol

The dealer is active only once at the beginning of the game, and during this phase of the game the players' inputs are prepared.

Communication between the players in our protocol will be restricted to sending messages to their neighbors in a complete binary tree.[2] First the players are assigned to nodes of a complete binary tree with n leaves. Each player is assigned to one leaf. Next the layer of internal nodes just above the leaves is assigned to players. This is done in such a way as to ensure that the player assigned to the parent node of two leaves is one of the two players assigned to those leaves. Then the remaining internal nodes are assigned arbitrarily to players who have not yet been assigned an internal node. Since there are $n-1$ internal nodes in a tree with n leaf nodes, we can ensure that no player is assigned to more than one internal node. One of the players will not be assigned to an internal node, and this will not matter.

Next, the dealer will select a player uniformly at random to be the "short player". The rest of the players will be "long players". It is important that the players *not be told* which player is the short player. This is necessary to avoid backwards induction as is discussed in Section 3.2.

Now the dealer independently samples three random variables X, Y and Z from a geometric distribution with parameter β. X will be the definitive iteration, or the round of the game in which the true secret is revealed. Y will be the amount of padding on the short player's input. Z will be the amount of additional padding on the long players' input. Thus, the short player will receive enough input to last for $X+Y$ rounds of the game, while the long players will receive enough input to last for $X+Y+Z$ rounds of the game.

The players' input will consist of several lists, with one list element for each round of the game. Thus the lists sent to the short player will be of length $X+Y$ and those sent to long players will be of length $X+Y+Z$. Each player i will receive the following four lists:

[2]Recall that a complete binary tree is a binary tree in which all the internal nodes have two descendants; all the leaves are at the two deepest levels; and the leaves on the deepest level are as far left as possible.

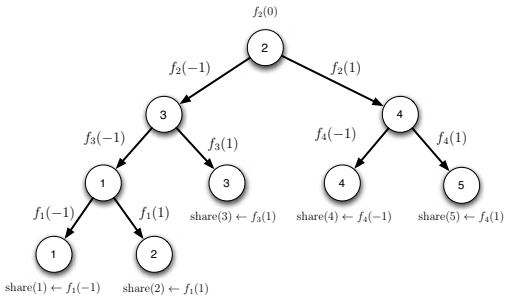

Figure 1: Example run of the dealer's protocol for a fixed round t. The function f_i is defined as $f_i(x) = a_i x + b_i$ where a_i and b_i are elements chosen independently and uniformly at random from the field $\mathbb{F}_q$. When $t = t^*$, b_2 is fixed at 0 and a_i is chosen uniformly at random; this ensures that $f_2(0)$ will be 0. When $t \neq t^*$, b_2 is chosen uniformly at random from $\mathbb{F}_q - 0$, which ensures that $f_2(0) \neq 0$.

- A list L_i consisting of elements of $\mathcal{S}$, or potential secrets. The true secret will be at index X of the list.
- A list S_i containing the shares of the indicator sequence for player i (in his role as a leaf of the tree.)
- For each node w (including leaves) with which i is associated, player i will receive a list T_i^w of pairs of tags (one for the *Up-Stage*, one for the *Down-Stage*) which will be used in the tag and hash scheme described below.
- For each node w with which i is associated, for each neighbor w' of w, player i will receive a list $H_i^{w,w'}$ with verification functions to check the authenticity of messages from w'.

The dealer's next task is to populate these lists. For simplicity of exposition, we assume that all the lists are created to be of length $X + Y + Z$, and that the short player's lists are truncated to length $X + Y$ later, before being sent to him. The dealer will fill the potential secret-lists L_i with elements chosen independently and uniformly at random from the set $\mathcal{S}$. These lists have the property that the true secret is in position X of the list.

Let q be a (fixed) large prime. The indicator sequence and tags will all be elements of the field $\mathbb{F}_q$ and all of the algebraic operations described hereafter will be over $\mathbb{F}_q$.

Next, to populate the shares-lists S_i, for each round $t \neq X$, the dealer will set h_t to be a uniformly random non-zero element of $\mathbb{F}_q$. h_X will be set to 0. Then for each t the dealer will create *iterated shares* of h_t as follows. First, the value h_t is assigned to the root of the tree. Next, for each node w of the tree for which a value v_t^w has been assigned, if w is a leaf, assign the value v_t^w to be the share of the corresponding player for round t. Otherwise, pick a random slope $\mu \neq 0$ and let f be the line with slope μ and y-intercept v_t^w (so that $f(0) = v_t^w$). Assign values $f(-1)$ and $f(1)$ to the left and right children of w, and recurse. The formal algorithm for this process is given in Algorithm 2, and an example run of this algorithm is illustrated in Figure 1.

The dealer's remaining task is to populate lists that implement the tag and hash verification scheme that will be used by the players. For each round t and each internal node w of the tree, the dealer generates tags g_t^w and $\bar{g}_t^w$ which are elements chosen independently and uniformly at

random from $\mathbb{F}_q$. These elements are put into the appropriate lists T^w. The tag g_t^w will be used on the *Down-Stage* of the protocol for the players that is described in Section 2.2, and $\bar{g}_t^w$ will be used in the *Up-Stage* of this protocol.

Now, for each round t and each node w, let $p(w)$ denote the parent of w, and let $\ell(w)$ and $r(w)$ be the left and right children of w if any. The dealer samples uniformly random non-zero elements a_t^w and b_t^w from $\mathbb{F}_q$ and sets

$$c_t^w = a_t^w * h_t + b_t^w * g_t^{p(w)}$$

The triple (a_t^w, b_t^w, c_t^w) consists of the verification function for the *Down-Stage* round of t for the node w. Similarly, the dealer generates such verification triples (a, b, c) to check the validity of the messages $(v_t^{\ell(w)}, \bar{g}_t^{\ell(w)})$ and $(v^{r(w)}, \bar{g}_t^{r(w)})$ that are to be received by w in the *Up-Stage*. These verification function triples are put into the appropriate lists H^w. (See Algorithm 3.) Note that for any triple $(a, b, c) \in \mathbb{F}_q^3$ with $a, b \neq 0$ there are q pairs $(x, y) \in \mathbb{F}_q^2$ which satisfy $ax + by = c$, so the values of the message and tag cannot be guessed from the verification function, except with probability $1/q$.

The dealer's task is now almost done. All that remains is to truncate the short player's lists and communicate the input to the players. The formal description of the entire protocol for the dealer is given in Algorithm 1.

2.2 Player's Protocol

The players' protocol consists of two stages: the *Up-Stage* and the *Down-Stage*. In the *Up-Stage* for round t each player i will send his share $S_i[t]$ along with the tag $\bar{g}_t^i$ up the tree to the parent node of the leaf with which player i is associated. Each internal node w receives two shares and tags from its two children and after checking their validity with the provided verification function $(H_t^{w,\ell(w)})$, uses these to reconstruct a line f, where the non-tag part of the messages received from the left and right children respectively represent $f(-1)$ and $f(1)$. Once f is reconstructed, w will send his "share", $f(0)$ along with his tag $\bar{g}_t^w$ to his parent, up the tree, who will recursively do the same thing. This procedure continues recursively up the tree until the root is reached. At the root, the function f, reconstructed from the shares of the two children, when evaluated at 0 will yield the indicator for the current round. In particular, the root sets $h_t = f_{\text{root}}(0)$. We next move into the *Down-Stage*.

In the *Down-Stage* for round t the root will send h_t (determined by the end of the *Up-Stage*) and its tag g_t^{root} to both of its children. Now each child node w, receiving from its parent $p(w)$ the pair $(h_t, g_t^{p(w)})$, will first check this message for consistency using its verification function (a_t^w, b_t^w, c_t^w). Once the message is verified, w, if it is itself an internal node, will recursively append its own tag to h_t and send the pair (h_t, g_t^w) to its children. If w is a leaf, then if $h_t = 0$ then $t = X$ and the player associated with w will output the secret for the current round as the true secret, and quit the protocol. Otherwise the next round begins.

If at any time in the protocol, a player detects deviation from the protocol by another player, then the player immediately outputs the potential secret for the current round as the true secret and quits the protocol. This detection of deviation could be in the form of no message being received when a message was supposed to be sent, or in the form of messages received that do not satisfy the verification function. Note that detection of cheating propagates to all the players, since once a player detecting cheating quits the protocol, everyone who was expecting messages from him will also fail to get these messages. The formal description of the protocol for the players is given in Algorithm 4, and an example run of this algorithm is illustrated in Figure 2.

Algorithm 1 *Dealer's Protocol*

n players, $\mathcal{S}$: set of potential secrets, $\mathbb{F}_q$: field of size q. $\beta \in (0,1)$: geometric distribution parameter.

1. Create a complete binary tree with n leaf nodes. Assign players to leaf nodes from left to right. Assign players to parents of leaf nodes by choosing the player assigned to one of the children on the node. Assign all other internal nodes to players who have not yet been assigned to an internal node, top down and left to right. Thus each player is assigned to a leaf node, and each player except one, is assigned to an internal node, and no two internal nodes are assigned to the same player.

2. Choose X, Y, and Z independently from a geometric distribution with parameter β.

3. Create the following lists for each player i:
   - List L_i for potential secrets
   - List S_i for shares of the indicator sequence.
   - For each node w with which i is associated, list T_i^w for tags.
   - For each node w with which i is associated and each neighbor w' of w, list $H_i^{(w,w')}$ for verification functions.

4. (Populate the lists) For each round $t \leq X + Y + Z$:
   (a) if $t = X$, $\sigma \leftarrow$ true secret.
   otherwise $\sigma \leftarrow$ random element from $\mathcal{S}$
   Add σ to all the lists L_i at position t.
   (b) if $t = X$, $h_t \leftarrow 0$
   otherwise $h_t \leftarrow$ random non-zero element from $\mathbb{F}_q$.
   (c) For each node w of the tree, create a 'share' v_t^w (to be set by *RecursiveShares*, subroutine 2)
   (d) Call *RecursiveShares*(root, h_t)
   (e) For each leaf w add v_t^w to the shares list S_i of corresponding player i at position t.
   (f) For each node w in the binary tree choose $(g_t^w, \bar{g}_t^w)$ uniformly at random from $\mathbb{F}_q^2$ and add it to T^w at position t
   (g) For node each w in the binary tree:
      i. if w has a parent $p(w)$:
      $V_p \leftarrow$ *VerificationFunction*$(h_t, g_t^{p(w)})$
      Add V_p to list $H^{w,p(w)}$ at position t.
      ii. if w has a left child $\ell(w)$:
      $V_\ell \leftarrow$ *VerificationFunction*$(v_t^{\ell(w)}, \bar{g}_t^{\ell(w)})$
      Add V_ℓ to list $H^{w,\ell(w)}$ at position t.
      iii. if w has a right child $r(w)$:
      $V_r \leftarrow$ *VerificationFunction*$(v_t^{r(w)}, \bar{g}_t^{r(w)})$
      Add V_r to list $H^{w,r(w)}$ at position t.

5. Choose a player i uniformly at random to be the short player. Truncate each list (L_i S_i, and multiple tag and hash lists) associated with player i to be length $X + Y$ *i.e.* delete the last Z elements of each such list. (All other players' lists remain of length $X + Y + Z$.)

6. Send the data to the appropriate players

Algorithm 2 *RecursiveShares* (node w , integer y):

1. $v_t^w \leftarrow y$.
2. If w has children $\ell(w)$ and $r(w)$:
   (a) Choose random slope μ.
   (b) Let f be the line with slope μ and y-intercept y.
   (c) *RecursiveShares*$(\ell(w), f(-1))$.
   (d) *RecursiveShares*$(r(w), f(1))$.

Algorithm 3 *VerificationFunction* ($\mathbb{F}_q$-element v, $\mathbb{F}_q$-element g): // v is the message, g is the tag

1. Choose a and b uniformly at random $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$
2. $c = a * v + b * g$.
3. return (a, b, c).

Algorithm 4 *Player's Protocol* for Player i

On round t:

Up-Stage:

1. Send $(S_i[t], \bar{g}_t^i)$ to parent in the tree.
2. If player i is also internal node w, then:
   (a) Receive $(v_t^{\ell(w)}, \bar{g}_t^{\ell(w)})$ from left child and check using verification function $H_t^{w,\ell(w)}$. If no message is received or message does not satisfy verification function, output $L_i[t]$ and QUIT. Otherwise set $f(-1) \leftarrow v_t^{\ell(w)}$.
   (b) Receive $(v_t^{r(w)}, \bar{g}_t^{r(w)})$ from left child and check using verification function $H_t^{w,r(w)}$. If no message is received or message does not satisfy verification function, output $L_i[t]$ and QUIT. Otherwise set $f(1) \leftarrow v_t^{r(w)}$.
   (c) Reconstruct a degree 1 polynomial f from $(-1, f(-1))$ and $(1, f(1))$.
   (d) if w is not the root, $v_t^w \leftarrow f(0)$
   Send $(v_t^w, \bar{g}_t^w)$ to parent in the tree.
   (e) Else if w is the root $h_t \leftarrow f(0)$.

Down-Stage:

1. If player i is the root: send (h_t, g^{root_t}) to left and right children.
2. Else if i is node w (including leaves):
   (a) Receive $(h_t, g_t^{p(w)})$ from parent $p(w)$. If no message received output $L_i[t]$ and QUIT.
   (b) Use verification function $H^{(w,p(w))}[t]$ to check validity of message. If message does not satisfy verification function, output $L_i[t]$ and QUIT.
   (c) If w is an internal node, then send indicator (h_t, g_t^w) to left and right children.
   (d) If $h_t = 0$, output $L_i[t]$ and QUIT.
3. $t \leftarrow t + 1$

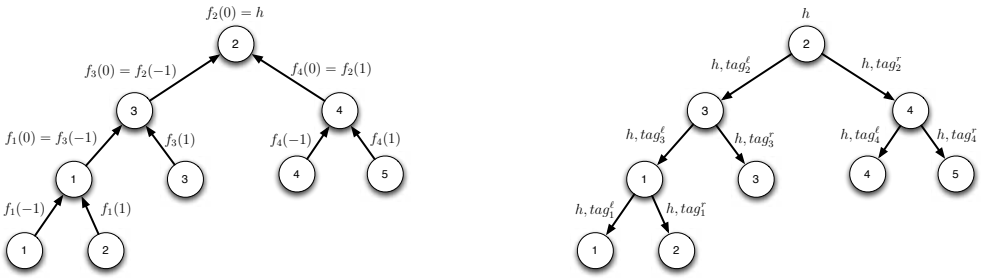

Figure 2: On the left is the up stage of the algorithm; on the right is the down stage.

3 Analysis of our Protocol

There are two ways in which a player may try to deviate from the protocol. First, he may try to remain in the protocol and send messages not specified by the protocol, or "fake" messages, in order to confuse other players. Second, he may simply output a value for the secret and leave the protocol early. As the following proposition shows, the tag-and-hash verification scheme used in the protocol makes it hard for a player to successfully fool other players by sending fake messages.

Proposition 3.1. *The probability that a faked message will satisfy the verification function is* $\frac{1}{q-1}$

Proof. Suppose a player A is to send a message v with tag g to player B who has a verification triple (a, b, c) such that $av + bg = c$. Suppose further that A wants to send a fake message instead. Note that A has no information about the verification function except that the message he is supposed to send satisfies it. In fact, recall that the verification function was created by the dealer by randomly selecting a and b and setting c to the appropriate value. Consider the following two cases.
Case 1: A wants to send a particular fake message v'. In this case, there is a unique $g' = (c - av')/b$ such that the pair (v', g') satisfies the verification function. With no knowledge of a or b, A's chance of correctly guessing g' is $\frac{1}{q-1}$.
Case 2: A just wants to send a fake message, but does not care what it is. In this case, there are exactly q pairs (v', g') which satisfy the verification function; one of these is (v, g). Thus only $q - 1$ fake pairs work. On the other hand, there are $(q-1)^2$ pairs of elements in $\mathbb{F}_q^2$ which are not of the form $(v, *)$ or $(*, g)$. So again, with no information about a or b, A's chance of guessing a pair that works is $\frac{1}{q-1}$. □

We will next consider the second type of deviation from the protocol, where a player deviates by leaving the protocol early. A player may consider deviating in this way if he either knows the secret with certainty, or with sufficiently high probability. The former only happens partway through the definitive iteration, when some of the players have learned this fact, but have not yet communicated it to the others. However, if a player decides not to transmit the fact that $h_t = 0$ to players lower in the tree, he does not gain anything by doing so. This is because the players who do not get the messages they are expecting will, in accordance with the protocol, output the secret for the current round, which is the correct secret. Thus, the only situation we still need to consider is whether a player can guess the position of the secret, with sufficiently high probability, before the definitive iteration. We deal with this situation in the next few lemmas.

Lemma 3.2. *A player deviating from the protocol cannot increase his expected payoff by more than* ϵ *unless his probability of successfully learning the secret by deviating is at least* $\frac{U - U_- + \epsilon}{U_+ - U_-}$

Proof. Let p be the threshold success probability required to make deviating worthwhile, ie to increase one's payoff by ϵ. Since the game ends when a player is caught deviating from the protocol (and certainly if the player stops sending messages, this *will* be detected by the other players), a failed attempt at cheating means that the player does not learn the secret. Thus a player's expected utility from cheating while everyone else follows the protocol is $pU_+ + (1-p)U_-$. On the other hand, following the protocol results in everyone learning the secret, with a utility of U. Thus, at the threshold for making cheating worthwhile, we have

$$pU_+ + (1-p)U_- = U + \epsilon$$

Solving this for p gives the desired result. □

We will now compute the probability of a player successfully guessing the secret before it is actually revealed during the protocol.

Lemma 3.3. *A player who initially received a list of length α has at most $\frac{2}{\alpha - t}$ chance of guessing the position of the secret on round t if it has not already been revealed.*

Proof. For $j \geq t$ let $P(\alpha, t, j)$ denote the probability that the secret is in the jth position in the list of potential secrets, given that the player initially received a list of length α, and $t-1$ rounds of the protocol have ended without the secret being revealed.

Let $|L_i|$ be the length of the list of potential secrets received by player i. Recall that X, Y and Z are geometric random variables corresponding to the position of the secret, the amount of padding on the short player's list, and the amount of additional padding on the long players' lists respectively. Then $|L_i| = X + Y$ if i is the short player, and $|L_i| = X + Y + Z$ if i is a long player. Let ξ_{short} be the event that the player was chosen as the short player and ξ_{long} be the event that the player was chosen as the long player. It follows that

$$\begin{aligned}
P(\alpha,t,j) &= \Pr(X = j | X \geq t, |L_i| = \alpha) \\
&= \Pr(X = j | X \geq t, X + Y = \alpha) \Pr(\xi_{short}) + \Pr(X = j | X \geq t, X + Y + Z = \alpha) \Pr(\xi_{long}) \\
&= \frac{\Pr(X = j, X \geq t, X + Y = \alpha)}{\Pr(X \geq t, X + Y = \alpha)} \cdot \frac{1}{n} + \frac{\Pr(X = j, X \geq t, X + Y + Z = \alpha)}{\Pr(X \geq t, X + Y + Z = \alpha)} \cdot \frac{n-1}{n} \\
&= \frac{\Pr(X = j, Y = \alpha - j)}{\sum_{x=t}^{\alpha-1} \Pr(X = x, Y = \alpha - x)} \cdot \frac{1}{n} + \frac{\Pr(X = j, Y + Z = \alpha - j)}{\sum_{x=t}^{\alpha-2} \Pr(X = x, Y + Z = \alpha - x)} \cdot \frac{n-1}{n} \\
&= \frac{\Pr(X = j) \Pr(Y = \alpha - j)}{\sum_{x=t}^{\alpha-1} \Pr(X = x) \Pr(Y = \alpha - x)} \cdot \frac{1}{n} + \frac{\Pr(X = j) \Pr(Y + Z = \alpha - j)}{\sum_{x=t}^{\alpha-2} \Pr(X = x) \Pr(Y + Z = \alpha - x)} \cdot \frac{n-1}{n} \\
&= \frac{(1-\beta)^{j-1}\beta(1-\beta)^{\alpha-j-1}\beta}{\sum_{x=t}^{\alpha-1}(1-\beta)^{x-1}\beta(1-\beta)^{\alpha-x-1}\beta} \cdot \frac{1}{n} + \frac{(1-\beta)^{j-1}\beta \Pr(Y + Z = \alpha - j)}{\sum_{x=t}^{\alpha-2}(1-\beta)^{x-1}\beta \Pr(Y + Z = \alpha - x)} \cdot \frac{n-1}{n} \\
&= \frac{(1-\beta)^{\alpha-2}\beta^2}{(\alpha - t)(1-\beta)^{\alpha-2}\beta^2} \cdot \frac{1}{n} + \frac{(\alpha - j - 1)(1-\beta)^{\alpha-2}\beta^3}{\sum_{x=t}^{\alpha-2}(\alpha - x - 1)(1-\beta)^{\alpha-2}\beta^3} \cdot \frac{n-1}{n} \\
&= \frac{1}{n(\alpha - t)} + \frac{(n-1)(\alpha - j - 1)}{n\sum_{k=1}^{\alpha-t-1} k} \\
&= \frac{1}{n(\alpha - t)} + \frac{2(n-1)(\alpha - j - 1)}{n(\alpha - t)(\alpha - t - 1)}
\end{aligned}$$

Recall that if player i received a list of length α at the beginning, and $t-1$ rounds of the protocol did not reveal the secret, then if player i guesses that the true secret is in position j, $P(\alpha, t, j)$ represents the probability that this guess is correct. For $j \geq t+1$ this also represents the probability that player can succesfully cheat by outputting the potential secret at position j and dropping out of the protocol. The situation is slightly different for $j = t$ because if the player is caught cheating on round t, all other players following the protocol will output the secret at position j so if that is correct, player i will gain no advantage from having guessed it. Thus, on round t the best probability of a successful cheat is obtained by guessing that the secret is in position $t+1$. In this case, the success probability is $\frac{1}{n(\alpha-t)} + \frac{2(n-1)(\alpha-t-2)}{n(\alpha-t)(\alpha-t-1)} < \frac{2}{\alpha-t}$ as asserted. □

3.1 Proof of Theorem 1.1

We now present the proof of Theorem 1.1

Proof. By Proposition 3.1 and Lemma 3.2, if $q > 1 + \frac{U_+ - U_-}{(U - U_- + \epsilon)}$ then no player can gain ϵ or more by faking messages in the protocol.

By Lemma 3.3 and Lemma 3.2, to prevent cheating by guessing the secret and prematurely exiting the protocol, it suffices if

$$\frac{2}{\alpha - t} < \frac{U - U_- + \epsilon}{U_+ - U_-}$$

Now, we know that $t < X$ and $\alpha - t$ is at least the short player's padding, and thus $\alpha - t > Y$. Thus it suffices to have $Y > \frac{U_+ - U_-}{2(U - U_- + \epsilon)}$. But Y is a geometric random variable with parameter β, so we require that

$$\Pr\left(Y > \frac{U_+ - U_-}{2(U - U_- + \epsilon)}\right) = (1-\beta)^{\frac{U_+ - U_-}{2(U - U_- + \epsilon)}} \tag{1}$$

Now, if we set

$$\beta = 1 - (1-\delta)^{\frac{2(U - U_- + \epsilon)}{U_+ - U_-}}$$

we have that $\Pr(Y > \frac{U_+ - U_-}{2(U - U_- + \epsilon)}) = 1 - \delta$ so that by Lemma 3.2, Lemma 3.3 and equation 1, with probability at least $1 - \delta$ no player can gain more than ϵ in expectation by guessing the position of the secret before the definitive iteration. It follows that with probability at least $1 - \delta$ the protocol is an ϵ-Nash equilibrium.

We now analyze the resource costs of our protocol. The communication tree has $2n - 1$ nodes. Each player is mapped to one leaf and at most one internal node. Players only communicate with their neighbors in the tree. So on each round, in the *Up-Stage* player i acting as a leaf, sends two field elements (a share and a tag) up to his parent, and then two more acting as an internal node. In the *Down-Stage*, acting as an internal node he sends two field elements each to his left and right child. Thus each player sends at most eight field elements per round, and thus the expected number of messages sent is $O(1)$ and the expected number of bits sent is $O(\log |\mathcal{S}|)$. Since the expected number of rounds is constant and the tree has $O(\log n)$ height, it follows that the expected latency is $O(\log n)$. □

3.2 A Note on Backwards Induction

The *backwards induction* problem arises when a multi-round protocol has a last round number that is known to all players. In particular, if it is globally known that the last round of the protocol is ℓ, then on the ℓ-th round, there is no longer any fear or reprisal to persuade a player to follow the protocol. But then if no player follows the protocol in the ℓ-th round, then in the $(\ell - 1)$-th round,

there is no reason for any player to follow the protocol. This same logic continues backwards to the very first round.

The backwards induction problem can occur with protocols that make cryptographic assumptions, since there will always be some round number, ℓ, in which enough time has passed so that even a computationally bounded player can break the cryptography. Even though ℓ may be far off in the future, it is globally known that the protocol will end at round ℓ, and so by backwards induction, even in the first round, there is no incentive for a player to follow the protocol.

As in [8], we protect against backwards induction by having both long and short players. As the above analysis shows, if β is chosen sufficiently small, we can ensure that Y will be large enough so that the probability of making a correct guess as to when the protocol ends is too small to enable profitable cheating for any player. Thus, even when a player gets to the second to the last element in all its lists, it can not be very sure that the protocol will end in the next round. All players are aware of these probabilities at the beginning of the protocol, and thus each player knows that no other player will be able to accurately guess when the protocol ends.

4 Conclusions and Future Work

We have presented *scalable* mechanisms for rational secret sharing problems. Our algorithms are scalable in the sense that the latency and the number of bits sent by each player is at most polylogarithmic in the number of players. For the n-out-of-n rational secret sharing, we give a scalable algorithm that is an ϵ-Nash equilibrium to solve this problem. For t-out-of-n rational secret sharing where 1) $m = \theta(n)$; and 2) the set of active players is chosen independently of the random bits of the dealer, we give a scalable algorithm that is a Nash equilibrium and ensures for any fixed, positive ϵ that if 1) at least a $m/n + \epsilon$ fraction of the players are active, all players will learn the secret; and 2) if less than a $t/n - \epsilon$ fraction of the players are active, then the secret can not be recovered.

Several open problems remain. First, while our algorithms lead to a $\theta(n)$ multiplicative reduction in communication costs for rational secure multiparty computation (SMPC), the overall bandwidth for this problem is still very high. It is known that there are certain functions, for which rational SMPC can not performed in a scalable manner (for example, the parity function). However, we ask: Can we find classes of well-motivated functions for scalable SMPC is possible? This is related to our second open problem which is: Can we design scalable algorithms for simulating a class of well-motivated mediators? In some sense, this problem may be harder than the SMPC problem, since some types of mediators offer different advice to different players. In other ways, the problem is easier: a simple global coin toss is an effective mediator for many games. A final important problem is: Can we design coalition-resistant scalable algorithms for rational secret sharing?

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 *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 53–62. ACM, 2006.

[2] R. Geambasu, T. Kohno, A.A. Levy, and H.M. Levy. Vanish: Increasing data privacy with self-destructing data. In *Proceedings of the 18th conference on USENIX security symposium*, pages 299–316. USENIX Association, 2009.

[3] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[4] S. Gordon and J. Katz. Rational secret sharing, revisited. *Security and Cryptography for Networks*, pages 229–241, 2006.

[5] J. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, page 632. ACM, 2004.

[6] S. Izmalkov, S. Micali, and M. Lepinski. Rational secure computation and ideal mechanism design. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 585–594. IEEE, 2005.

[7] V. King and J. Saia. Breaking the O (n 2) bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 420–429. ACM, 2010.

[8] G. Kol and M. Naor. Games for exchanging information. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 423–432. ACM, 2008.

[9] M. Lepinksi, S. Micali, and A. Shelat. Collusion-free protocols. In *ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING*, volume 37, page 543. Citeseer, 2005.

[10] M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely fair SFE and coalition-safe cheap talk. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 1–10. ACM, 2004.

[11] A. Lysyanskaya and N. Triandopoulos. Rationality and adversarial behavior in multi-party computation. *Advances in Cryptology-CRYPTO 2006*, pages 180–197, 2006.

[12] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[13] M.N. Wegman and J.L. Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

A Scalable m-out-of-n Secret Sharing

As discussed previously, it does not seem possible to design *scalable* algorithms for secret sharing in the case when either t is much smaller than n; or the subset of size t of active players may be chosen in a completely arbitrary manner. We now describe the situation where 1) $t < n$, but $t = \theta(n)$; and 2) the subset of active players does not depend on the random bits of the dealer.

We now sketch our algorithm. Since the algorithm is a variant on the n-out-of-n scheme, for conciseness, we describe here only the places where the two algorithms defer. First, the dealer creates a communication tree as illustrated in Figure 3. The internal nodes of the tree are now *supernodes*, which consist of $C \log n$ players chosen independently, uniformly at random and with replacement, for some constant C chosen sufficiently large (but depending only on t). As before, there is a separate leaf node for each player of the tree. Every internal node of height greater than 1 has exactly 2 children and every internal node of height 1 has $C \log n$ children. [3]

Dealer's Protocol: When creating the list S_i for each player i, the protocol for the dealer runs as before, until we reach the supernodes of height 1. Let $x = C \log n$ and $m' = xm/n$. Then at the supernodes of height 1, the dealer now uses Shamir m'-out-of-x secret sharing to create iterated shares for the x children. The algorithm for populating the other lists (S_i, T_i^w, and $H_i^{w,w'}$) is similar to that used for the n-out-of-n algorithm.

Player's Protocol: The protocol for the players consists of an upward and downward phase. In the upward phase, the players at the leaf nodes first send their iterated shares of f to all players in the supernodes above them. The players in these supernodes, use the shares received to reassemble the appropriate value for that supernode, check this value with the hash and tag scheme and then send the value up to the players in the supernode above them. This continues until the root node is reached, at which point, the value of the function f at 0 is reconstructed.

In the downward phase, the secret is sent down the tree from each parent supernode to its children, using all-to-all communication among the players in the supernodes, until at the bottom, all players learn the value of f for that round. The players now act on the information they receive (or don't receive) exactly as they do in the n-out-of-n algorithm.

Analysis Sketch: The correctness of this algorithm follows from two arguments. First, that if the algorithm is followed, then with high probability,[4] all players learn the secret. The idea to show this is to first let $f = (m + \epsilon)/n$. Then, apply Chernoff and then union bounds to show that with high probability, 1) all supernodes contain at least a m/n fraction of active players; and 2) all supernodes at height 1 receive at least a m' shares from active children. The second argument is that the algorithm is a Nash equilibria, and the argument for this is similar to the argument for the n-out-of-n case.

[3] In fact, for arbitrary n, this will have to be degree between $C \log n$ and $2C \log n$. For simplicity, we assume degree of $C \log n$.

[4] We use the phrase *with high probability* to mean with probability $1 - 1/n^k$ for any fixed constant k.

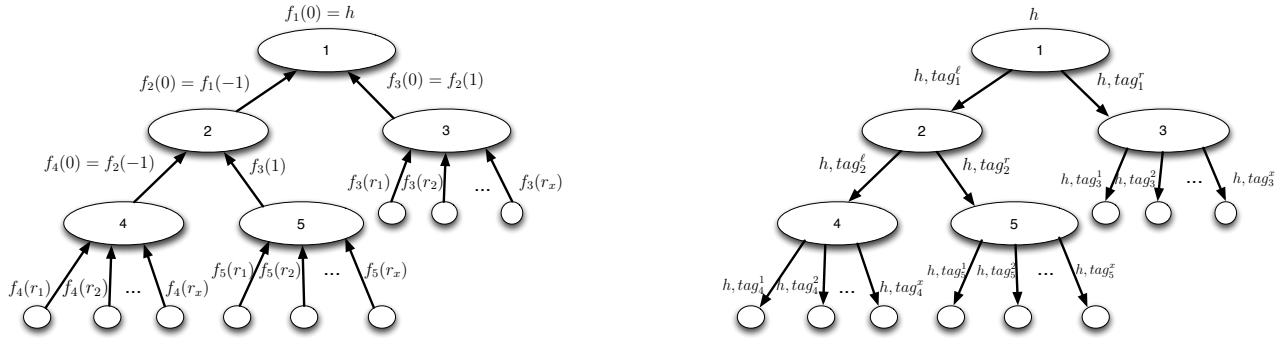

Figure 3: Illustration of the scalable m-out-of-n secret sharing algorithm. The internal supernodes (1, 2, 3, 4 and 5) each consist of $C \log n$ players selected independently and uniformly at random with replacement. The degree of each node with height 1 (i.e. nodes 4, 5, and 3) is $x = C \log n$. On the left is the up stage of the algorithm; on the right is the down stage.