Distrib. Comput. (2015) 28:171–187
DOI 10.1007/s00446-014-0233-4

Scalable mechanisms for rational secret sharing

Varsha Dani · Mahnush Movahedi · Jared Saia

Received: 18 April 2013 / Accepted: 25 October 2014 / Published online: 13 November 2014

Abstract We consider the classical secret sharing problem in the case where all agents are rational. In recent work, Kol and Naor show that, when there are two players, in the non-simultaneous communication model, *i.e.*, when rushing is possible, there is no Nash equilibrium that ensures both players learn the secret. However, they describe a mechanism for this problem, for any number of players, that is an ϵ-*Nash equilibrium*, in 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. We address this issue by describing mechanisms for rational secret sharing that are designed for large $n \geq 3$. Our first result is a non-cryptographic mechanism for n-out-of-n rational secret sharing that is Nash equilibrium, rather than just ϵ-Nash equilibrium. Our second result is a cryptographic mechanism for rational t-out-of-n secret sharing that is everlasting ϵ-Nash equilibrium. Both our mechanisms are *scalable* in the sense that they requires each player to send only an expected $O(\log n)$ bits. Furthermore, the latency of these mechanisms is $O(\log n)$ in expectation, compared to $O(n)$ expected latency for the Kol and Naor result. Both of our mechanisms are not susceptible to backwards induction.

This research was partially supported by NSF CAREER Award 0644058,NSF CCR-0313160, and an AFOSR MURI grant.

V. Dani · M. Movahedi (✉) · J. Saia
Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386, USA
e-mail: movahedi@cs.unm.edu

V. Dani
e-mail: varsha@cs.unm.edu

J. Saia
e-mail: saia@cs.unm.edu

Keywords Distributed scalable algorithm · Game theory · Rational secret sharing

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

1 Introduction

Secret sharing is one of the most fundamental problems in security, and is an important primitive in many cryptographic protocols. In t-out-of-n secret sharing, we are interested in designing a protocol to distribute shares of a secret to each n players and to reconstruct the secret ensuring that: (1) if any group of t players follow the protocol, they will all learn the secret; and (2) knowledge of less than t of the shares reveals nothing about the secret. In rational secret sharing, all players are rational. Moreover, we want our protocol to be a *Nash equilibrium* in the sense that no player can improve its utility by deviating from the protocol, given that all other players are following the protocol.

Recently, there has been interest in solving *rational secret sharing* [1,6,7,10,11]. 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.

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 $O(\log n)$ bits; and (2) has $O(\log n)$ expected latency. We also design a scalable mechanism for t-out-of-n rational secret sharing when the parties are computationally bounded.

1.1 The problem

Shares of a secret are to be dealt to n rational players, who will later reconstruct the secret from the shares. The players

are *learning-preferring,* in the sense that each player prefers every outcome in which he learns the secret to any outcome in which he does not learn the secret. However, a player may prefer the situation where he learns the secret and other players do not to the situation where all players learn the secret.

The secret is an arbitrary element of a large (fixed) finite field $\mathbb{F}_q$. At the beginning of the game, a dealer provides the shares to the players. The dealer has no further role in the game. The players must then communicate with each other in order to recover the secret.

Communication between the players is *point-to-point* and through secure 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. Communication is *synchronous* in that there is an upper-bound 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 his 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.

1.2 Related work

The problem of secret sharing with rational players was introduced by Halpern and Teague in [7]. They showed that, rational secret sharing is possible using randomized mechanisms with constant expected running time and it is impossible using a mechanism that has a fixed running time. They also show that there is no practical mechanism for 2-out-of-2 rational secret sharing even with an infinite game tree. A mechanism is practical if it is a Nash equilibrium that survives iterated deletion of weakly-dominated strategies. Since then, there has been significant work on the problem of rational secret sharing, including results of Gordon and Katz [6], Abraham et al. [1], Lysyanskaya and Triandopoulos [11], Asharov and Lindell [2], Kol and Naor [10] et al. [15] and Fuchsbauer et al. [4].

Most of this related work except for [2,4,10,15], assume the existence of simultaneous communication, either by broadcast or private channels. Several of the protocols proposed [1,6,11,15] require cryptographic assumptions and achieve an equilibrium under the assumption that the players are computationally bounded. In contrast, our results do not assume simultaneous communication and our n-out-of-n algorithm does not make any cryptographic assumptions.

Fuchsbauer et al. [4] proposed protocols for 2-out-of-2 and t-out-of-n rational secret sharing. Similarly to our model, they assume the players only have point-to-point channels and they do not require broadcast channel. Unlike us, the proposed protocols in [4] are based on cryptographic assumptions and the authors introduce the notion of a computational strict Nash equilibrium, that is a Nash equilibrium when all players are polynomial time bounded. They show that their protocols are both computational strict Nash. A caveat is that the protocols are susceptible to backwards induction.

Zhang, Tartary, and Wang in [15], propose a protocol for rational t-out-of-n secret sharing scheme based on the Chinese reminder theorem. Their protocol works in a communication model similar to our model. Their protocol requires a synchronous but not simultaneous broadcast channel and synchronous but not simultaneous point-to-point private channels. They assume some computational hardness related to the discrete logarithm problem and RSA. The proposed protocol is a $(t-1)$-resilient computational strict Nash equilibrium that is stable with respect to trembles. Their protocol runs in time polynomial in k and has shares of size $O(k)$ bits where k is the protocol security parameter.

The work of Kol and Naor [10] is closest to our own work and our protocols make use of several clever ideas from their result. 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, at least for the case of two players. They thus consider and solve the problem of designing an ϵ-Nash equilibrium for the problem in this communication model. An ϵ-Nash equilibrium is close to an equilibrium in the sense that no player can gain more than ϵ utility by unilaterally deviating from it. Furthermore, the equilibrium they achieve is *everlasting* in the sense that after any history that is consistent with all players following the protocol, following the protocol continues to be an ϵ-Nash equilibrium.

The impossibility of a Nash equilibrium for two players carries over to the setting with secure private channels, since there is no difference between private channels and broadcast channels when there are only two players. However, one might hope that the algorithm of [10] could be efficiently simulated over secure private channels to give an everlasting ϵ-Nash equilibrium. Unfortunately, simulation of broadcast over private channels is expensive. To the best of our knowledge, a single broadcast requires each player to send $\Theta(n)$ messages.

In [3], Dani et al. overcame this difficulty, by providing a *scalable* protocol for rational secret sharing, in which each player only sends $O(1)$ bits per round and the expected number of rounds is constant (although each round takes $O(\log n)$ time). Moreover, following the protocol is an ϵ-Nash equilibrium. Unfortunately, a certain bad event with small but constant probability caused some players, when they recog-

nized it, to deviate from the protocol so that the equilibrium from [3] is not everlasting.

1.3 Our results

The main result of this paper is presented as Theorem 1.

Theorem 1 *Let $n \geq 3$. There exists a protocol for rational n-out-of-n secret sharing with the following properties.*

- *The protocol is a Nash equilibrium in which all players learn the secret.*
- *In expectation, the protocol requires each player to send $O(\log n)$ bits, and has latency $O(\log n)$.*
- *The protocol is not robust to coalition on any size.*

This paper is the full version of the extended abstract in [3]. Additionally, it improves on the work in [3] in two ways. First, the new proposed algorithm for n-out-of-n secret sharing is correct with probability one and there is no probability of error. Second, we show that our new protocol for n-out-of-n is a Nash equilibrium, not just an ϵ-Nash equilibrium, as long as $n \geq 3$.

t-out-of-n secret sharing We also consider the problem of t-out-of-n rational secret sharing for the case where $t < n$. Scalable rational secret sharing for the t-out-of-n case may also be of interest for applications like the Vanish peer-to-peer system [5]. Vanish uses secret sharing in a peer-to-peer system with churn in order to provide data that vanishes over time. In this setting we prove the following.

Theorem 2 *Let $n \geq 3$ and $t \leq n$. Assuming the players are computationally bounded to polynomial-time algorithms, there exists a protocol for rational t-out-of-n secret sharing with the following properties.*

- *The protocol is an everlasting ϵ-Nash equilibrium in which all active players learn the secret, where ϵ can be any arbitrary positive value.*
- *In expectation, the protocol requires each player to send $O(\log n)$ bits, and has latency $O(\log n)$.*
- *The protocol is not robust to coalition of any size.*

This is an improvement to the $\Theta(n)$-out-of-n result we proved in [3], in the sense that in this new result, t can be any number smaller than n. However, in our new t-out-of-n protocol, we require a cryptographic digital signature scheme with security parameter $\kappa = \max_j \frac{U_j^+ + U_j}{2U_j + \epsilon}$. See Sect. 2.2 for definitions of U_j^+ and U_j.

1.4 Our approach

The difficulty in designing a Nash equilibrium in a communication model where rushing is possible, is that the last player to send out his share has no incentive to actually do so. He already has the shares of all the other players and can recover the secret alone. To get around this, it is common (see [1,6,7,10,11]) for the protocol to have a number of fake rounds designed to catch cheaters. The uncertainty in knowing which is the "definitive" round, during which the true secret will be revealed causes players to cooperate.

In the work of [10] this uncertainty is created by dealing one player only enough data to play until the round preceding the definitive one. Thus, there is a single "short" player and $n-1$ "long" players. None of the players know whether they are short or long. The long players must broadcast their information every round, since they cannot predict the definitive round in advance. The short player knows the definitive round in advance, but has no information about the secret in that round. In the definitive round the short player is the last to speak so that he (and all the other players) receives the shares of all the long players and can recover the secret. The short player's failure to broadcast a message is what cues the other players to the end of the game, and they too can recover the secret.

Moreover, having learned the secret, the short player cannot pretend that he actually had a share for that round as the messages sent by all the players are verified by a tag and hash scheme (see, e.g., [10,12,14]). In fact, it is the small but positive chance of cracking the tag and hash scheme that results in the protocol from [10] being an ϵ-Nash equilibrium rather than a Nash equilibrium. In this paper for the n-out-of-n case, we overcome this problem by ensuring there are at least two short players when $n \geq 3$. Thus, our protocol is a Nash equilibrium for this case since if a single short player breaks the tag and hash scheme, he cannot prevent any other player from learning the secret. See Sect. 4 for complete analysis.

Unfortunately, the same trick of having more than one short player to achieve a Nash-equilibrium cannot be easily applied to the t-out-of-n case. Since the dealer is not aware of the identity of the active players, he cannot choose short players in a way to make sure there are at least two short players active in the game. Since there is a chance that there is only one active short player, a short player will now have some incentive to try to forge the tag and hash scheme.

Due to the using of short and long players, both of our protocols are susceptible to coalitions of any size. Consider a long player who is the first one that learns the secret. He can join together with a short player to learn that this round is definitive. Thus, both players in this coalition will learn the secret early, and the remaining players will be fooled into thinking that the previous round was the definitive round.

We introduce two novel techniques to ensure scalable communication. The first technique is to arrange players at the leaves and nodes of a complete binary tree, and require that the players only communicate with their neighbors in the tree. The assignment of players to the leaves is independently random in every round, and their assignment to internal nodes is related to their assignment to leaves according to a predefined scheme. Every round of the game, information travels up to the root where it is decoded and then travels back down again to the leaves. We also use short and long players to determine the definitive round. The short players are the players who are parents of the leaves in the definitive round and all other players are long players. So, every player is a child of a short player in the definitive round. In the definitive round, the short players will fail to send messages to all the leaf nodes. Thus, every player, upon not receiving a message, will learn that this is the definitive round and so will learn the secret.

The second main idea is that we make use of an iterated secret sharing scheme over this tree in order to divide up shares of secrets among the players. This scheme is similar to that used in recent work by King and Saia [8] on the problem of scalable Byzantine agreement, and suggests a deeper connection between the two problems.

One drawback of our both protocols is vulnerability to coalitions of any size. The player who is at the root in the definitive round can team up with a short player and learn the secret early without telling it to others.

As in previous work [10,12,14] we use a information-theoretic message authentication codes (also referred as tag-and-hash scheme) to ensure that players cannot forge messages in the protocol in our n-out-of-n protocol.

1.5 Paper organization

The rest of this paper is laid out as follows. In Sect. 2, we give notation and preliminaries. In Sect. 3, we describe our algorithm for scalable n-out-of-n secret sharing. In Sect. 4, we analyze this algorithm; the main result of this section is a proof of Theorem 1. In Sect. 5, we give our algorithm and analysis for scalable t-out-of-n secret sharing; the main result of this section is a proof of Theorem 2. Finally in Sect. 6, we conclude and give directions for future work.

2 Notation and preliminaries

The secret to be shared is an arbitrary element of a set $\mathcal{S}$. There are n players with distinct player ids in $[n] = \{1, 2, \ldots n\}$. During the course of the algorithm, we will want to do arithmetic manipulations with player ids and shares of the secret, including adding in, or multiplying by random elements to preserve secrecy. In order to be able to do these sorts of manipulations, we embed the sets $\mathcal{S}$ and $[n]$ into a finite field $\mathbb{F}$ of size $q > \max\{n, |\mathcal{S}|\}$. The latter embedding will be the canonical one; the former may be arbitrary, but is assumed to be known to all parties.

The messages sent by players in the algorithm will be elements of $\mathbb{F}$. The length of any such message is $\log q = \Omega(\log n)$. Since our goal is to provide a scalable algorithm we cannot afford the message lengths to be much bigger than that. We will choose $\mathbb{F}$ to be a prime field of size $q = O(n)$. We remark that although generally $\mathcal{S}$ is of constant size, we can tolerate $|\mathcal{S}| = O(n)$.

2.1 Utility functions

We will denote the utility function of player j by u_j. As mentioned before, we assume that the players are learning preferring, *i.e.*, each player prefers any outcome in which he learns the secret to every outcome in which he does not learn the secret. More formally, for outcome $\mathbf{o}$ of the game, let $R(\mathbf{o})$ denote the set of players who learn the secret. If $\mathbf{o}$ and $\mathbf{o}'$ are outcomes of the game such that $j \in R(\mathbf{o}) \setminus R(\mathbf{o}')$, then $u_j(\mathbf{o}) > u_j(\mathbf{o}')$. As in [10], we denote

$$U_j^+ = \max\{u_j(\mathbf{o}) \mid j \in R(\mathbf{o})\}$$
$$U_j = \min\{u_j(\mathbf{o}) \mid j \in R(\mathbf{o})\}$$
$$U_j^- = \max\{u_j(\mathbf{o}) \mid j \notin R(\mathbf{o})\}.$$

Thus U_j^+ is the utility to player j of the best possible outcome for j, U_j is the utility to j of the worst possible outcome in which j still learns the secret, and U_j^- is the best possible utility to j when he does not learn the secret. By the learning-preferring assumption, we have for all j,

$$U_j^+ \geq U_j > U_j^-.$$

We will denote by $\mathcal{U}$, the quantity

$$\mathcal{U} := \max_{j \in [n]} \frac{U_j^+ - U_j^-}{U_j - U_j^-}.$$

Note that $\mathcal{U} \geq 1$. We assume that $\mathcal{U}$ is constant, *i.e.*, that it does not depend on n.[1]

Similar to previous work on rational secret sharing [10], we also assume that the utilities are such that the players have an incentive to play the game rather than just guess the secret at random. Moreover, as in previous works, we assume that the prior distribution of the secret is uniform for all players. Thus, if player j decides to guess the secret, with probability $1/|\mathcal{S}|$ he can get at most the maximum utility of U_j^+ and with probability of $1 - 1/|\mathcal{S}|$ can get at most the maximum utility

[1] Technically, we can achieve scalable (polylog) communication even if we allow $\mathcal{U}$ to be as big as polylog(n).

of U_j^-. Thus, player j's utility is at most $\frac{U_j^+-(1-|\mathcal{S}|)U_j^-}{|\mathcal{S}|}$ if he guesses instead of running the protocol. On the other hand, if the protocol runs correctly, he has utility at least U_j. Hence, we require $\frac{U_j^+-(1-|\mathcal{S}|)U_j^-}{|\mathcal{S}|} < U_j$ for all $1 \leq j \leq n$. To satisfy these inequalities we need: $|\mathcal{S}| > \max_{j\in[n]} \frac{U_j^+-U_j^-}{U_j-U_j^-}$. Thus we require,

$$\mathcal{U} < |\mathcal{S}|. \tag{1}$$

2.2 Game theoretic concepts

In this section, we review a few game-theoretic background concepts. We adapt the standard concepts here from the previous work (e.g., see [1,7]). We describe a game with a tree, where each node represents the local states of players based on their moves in previous rounds. The local state of player j describes player j's initial information and messages sent and received by player j. At each round, all the players move to a new state by deciding which messages to send. This decision can happen after receiving other players' messages. Each message takes exactly one round to arrive.

A strategy or protocol for player j is a function from player j's information set (*i.e.*, player j's local state) to actions. A strategy can be a randomized function. A joint strategy $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ is a tuple of strategies, one for each player. We let $\vec{\sigma} = (\vec{\sigma}_{-j}, \sigma_j)$, where $\vec{\sigma}_{-j}$ denotes a tuple consisting of each players strategy in $\vec{\sigma}$ other than player j's. $U_j(\vec{\sigma})$ is player j's expected utility if $\vec{\sigma}$ is played. Thus, the utility function depends on the path in the tree that starts from the root.

Recall that a joint strategies for a game is called a *Nash equilibrium* if no player has an incentive to unilaterally deviate from the equilibrium strategy, when all others are following it. Formally, $\vec{\sigma}$ is a Nash equilibrium if,

$$\forall j, \forall \sigma_j', U_j(\vec{\sigma}_{-j}, \sigma_j) \geq U_j(\vec{\sigma}_{-j}, \sigma_j').$$

In the same way, $\vec{\sigma}$ is an ϵ-Nash equilibrium if,

$$\forall j, \forall \sigma_j', U_j(\vec{\sigma}_{-j}, \sigma_j) + \epsilon \geq U_j(\vec{\sigma}_{-j}, \sigma_j').$$

In games of incomplete information which have multiple rounds, there is the further question of whether the players are forced to commit to their strategies before the start of the game or whether they have the option to change strategies in the middle of the game, after some rounds have been played and they may learn some new information. Kol and Naor [10] defined a equilibrium to be *everlasting* if after any history that is consistent with all players following the equilibrium strategy, it is still true (despite whatever new information the players may have learned over that history) that a player choosing to deviate unilaterally cannot gain in expectation, *i.e.*, following the prescribed strategy remains in equilibrium. For some equilibrium concepts such as ϵ-Nash, everlastingness is a stronger concept than the same equilibrium where the strategies are committed to up front [10].

However, when the underlying equilibrium is Nash, everlastingness does not make a difference. That is, an everlasting Nash equilibrium is the same as a Nash equilibrium. Nash equilibrium cannot fail to be an everlasting equilibrium. Consider a Nash equilibrium joint strategy $\vec{\sigma}$ and a path related to this joint strategy in the game tree. If this joint strategy is not an everlasting equilibrium, then along the path, there is some node consistent with $\vec{\sigma} = (\vec{\sigma}_{-j}, \sigma_j)$ at which some player, say player j, has an incentive to deviate and follow strategy σ_j'. Thus, the strategy σ_j' would dominate strategy σ_j, contradicting the assumption that the original $\vec{\sigma}$ is Nash.

Linger Avoiding Assumption: In this paper, we make the standard *linger avoiding assumption*. This means that our mechanisms do not require players to continue after they learn the secret. Without this assumption, there are trivial but fragile mechanisms that are Nash equilibria. See [10] for details.

2.3 Information-theoretic message authentication codes

A Message Authentication Code (MAC) scheme consists of three algorithms Key-Generation, Mac and Verification. The key-generation algorithm generates a key as its name suggests. The Mac algorithm takes a key and a message from a field as input and outputs a tag. The Verification algorithm takes a key, a message and a tag as input, and verifies the tag. In our algorithm we use the following information theoretic MAC scheme in field $\mathbb{F}$ with q elements. The Key-Generation algorithm outputs $c \in \mathbb{F}$ and $b \in \mathbb{F}^* = \mathbb{F} \setminus \{0\}$ independently, uniformly at random. (b, c) is the verification vector, to be given to the recipient of the message y. The Mac algorithm takes an $\mathbb{F}$-element message y and generates the tag, $a = c - b \cdot y$, which will be given to the sender of message y to send it along with the message. The recipient of the message y can run the Verification algorithm to verify the tag by testing if $c - b \cdot y$ is equal to a.

Next, we discuss the properties of the above information theoretic MAC scheme. This scheme makes it hard for a sender to successfully fool the intended recipient of a message by sending a faked message. At the same time, it does not give the recipient of the message any information about the message prior to receiving it. Such schemes have been used before (see *e.g.* [10,12,14]); we include the following proposition for completeness. See Lemma 1 of [12] for the proof of the following proposition.

Proposition 1 *The MAC scheme described above has the following properties:*

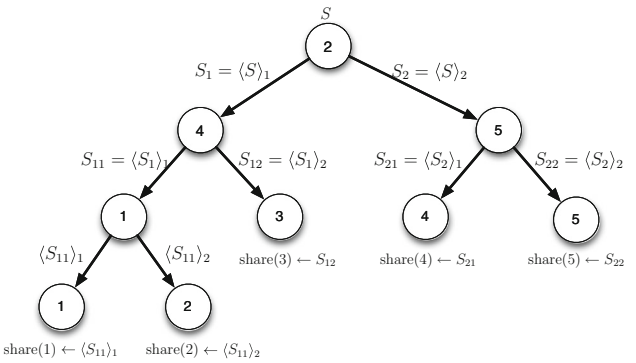

Fig. 1 Construction of the iterated shares. We define $\langle S \rangle_1$ as the first share and $\langle S \rangle_2$ as the second share generated from the 2-out-of-2 Shamir's secret sharing scheme

- *The verification vector contains no information about the message,* i.e.*, the probability of correctly guessing the message given the verification vector is the same as the unconditional probability of guessing the message.*
- *The probability that a faked message will satisfy the verification function is* $\frac{1}{q-1}$.

3 Algorithm for *n*-out-of-*n* secret sharing

We now describe our scalable mechanism for n-out-of-n secret sharing. First, in Sects. 3.1 and 3.2 we describe the communication tree and how to label it that is used by the dealer and players. An informal description of the mechanism follows in Sect. 3.3. The formal descriptions of the dealer's and players' protocols appear respectively as Algorithms 2 and 3.

3.1 The communication tree

Communication between the players in our protocol will be restricted to sending messages to their neighbors in a *communication tree*. The communication tree is a complete binary tree with n leaves. Recall that a complete binary tree is a binary tree in which all the internal nodes have exactly two descendants, all the leaves are at the two deepest levels, and the leaves on the deepest level are as far left as possible.

The communication tree is used to build *iterated shares* of a value. The dealer sends these iterated shares to the players, so they can reconstruct the value later. The iterated shares players receive are constructed by starting with the value to be reconstructed at the root and recursively constructing 2-out-of-2 Shamir shares down the tree, all the way down to the leaves. The shares at the leaves are the iterated shares the players receive. See Fig. 1 and Algorithm 1 for details of how the iterated shares are constructed. At reconstruction time, shares are sent up the tree. At each internal node, a pair of shares received from the two children is reconstructed into a degree 1 polynomial which is used to obtain the value to be sent further up the tree. At the root, the original symbol is reconstructed and transmitted down the tree. Note that the advantage of this scheme over simply using n-out-of-n Shamir shares is that the size of the messages does not increase as the messages are transmitted up the tree. Lemma 1 describes that shares resulted from Algorithm 1 is an n-out-of-n secret sharing scheme.

Algorithm 1 *RecursiveShares* (node w , $\mathbb{F}$-element y):

Parameters: V is a n-leaf complete binary tree global data structure; for node w, $V(w)$ denotes the location for the data associated with w.
Usage: Initially called with the root node and the value for which shares are to be created, this function populates V with intermediate values. The values at the leaves are the shares for the players at the corresponding leaves of the communication tree.

1. $V(w) \leftarrow y$.
2. If w has children $L(w)$ and $R(w)$:
   (a) share y using 2-out-of-2 secret sharing. Let the shares be y_L, y_R
   (b) *RecursiveShares*$(L(w), y_L)$.
   (c) *RecursiveShares*$(R(w), y_R)$.

Lemma 1 *Let* $\sigma \in \mathbb{F}$ *be a value that is encoded into n iterated shares,* $\sigma_1, \ldots, \sigma_n$*, by Algorithm 1. Then,* σ *can be decoded from all n of the shares, but knowledge of fewer than n of the shares reveals no information about* σ.

Proof Starting with the shares $\sigma_1, \ldots, \sigma_n$ at the leaves of the tree, we can recursively reconstruct the values bottom-up using 2-out-of-2 Shamir's secret sharing reconstruction algorithm. Since this is exactly the reverse of the process used to create these shares, the value at the root will be σ.

To see why fewer than n shares give us no information about σ, observe that the values at the two children of the root were created by 2-out-of-2 secret sharing algorithm. Both of these values together determine the secret, but a single one of them does not reveal any information about the secret. Thus the values at the children of the root *individually* contain no information about the value of the root, and in order to decode the value at the root, we need both the values at its children. But now, this reasoning applies recursively to all the internal nodes, relative to their children. Suppose there is a leaf of the tree at which the share is missing. Then, the share of its parent cannot be decoded because it is equally likely to be any element of the $\mathbb{F}$. This propagates up to its grandparent, and then its great-grandparent and so on all the way to the root, so that the root cannot be decoded. Thus, if even one of the shares is missing, the remaining shares provide no information about the value of σ. □

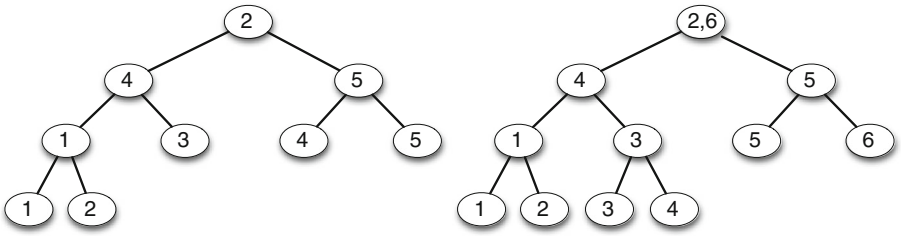

Fig. 2 Communication trees for five players (*left*) and six players (*right*)

3.2 Labeling the communication tree: short and long players

Recall that the short players are the players who are parents of the leaves in the communication tree in the definitive round and all other players are long players. To make the definition easier, we devise a specific labeling for the communication tree. We have exactly n labels and thus, the labels may be repeated in the tree. The leaves will be labeled 1 to n from left to right. Next every internal node which is a parent of two leaves is labeled with the odd label from among its two children. Finally, the remaining internal nodes are labeled in order with even numbers, proceeding top to bottom and left to right, starting with 2 at the root. If n is odd, then each even number appears at some internal node. If n is even, we will place the last even number, n at the root, along with 2, so the root will have two labels. Figure 2 illustrates the labeling scheme for five and six players. The tree thus labeled has the following properties:

1. Every even label occurs at some internal node. If n is odd, there will be an odd label that occurs only at a leaf and not at any internal node. This will not matter.
2. No even labeled internal node has an odd labeled node above it.
3. Every path from root to leaf has exactly one odd label (the same odd label may occur once or twice on the path).

We generate a permutation of players and assign players to n labels based on his permutation position. We use this permutation to assign players to the nodes of the tree. A player is assigned to a node if they have the same labels. Thus, each player can be assigned to multiple nodes of the tree and two players may assign to the root node of the tree. Based on this labeling, the short players are ones in odd positions in the permutation in the definitive round and all the other players are long players.[2]

As discussed in Sects. 3.4 and 4, it is critical in our analysis that the players themselves do not know which are short or long until the definitive round. Further, each player must have as a priori equal chance to be short or long. Thus, the permutation and the definitive round must be random and unknown to the players. Therefore, in our protocol, the dealer first generates random trees based on random permutations and chooses a definitive round randomly. Then, he assigns the long and short players accordingly to the choice of the tree in the definitive round.

3.3 Our algorithm

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

The dealer independently samples two random variables X and Y from a geometric distribution with parameter β (to be determined later). 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 long players' input. We have two kinds of players: short players will receive enough input to last for X rounds of the game while long players will receive enough input to last for $X + Y$ rounds of the game.

In order not to reveal which players are the short players, the players will be reassigned to new random positions in the tree in each round. This is accomplished by choosing a random permutation of the players and label them for each round. Again recall that the short players are the ones who are at odd labeled nodes in the definitive round.

Since the players must be at different nodes in the tree each round, their input must contain this information. At the same time, the positions of the players for all the rounds cannot be revealed up front, since this may give away information about who the short players are. A naive idea to solve this problem is, in each round, to distribute shares of the permutation for the next round. Then, during each round, the players could reconstruct the permutation from the shares and use it to reposition for the next round. Unfortunately, there is a problem with this approach. To represent permutations of n symbols, we need a field of size at least $n!$. To transmit elements of such a field, players would need to send messages of length $\log(n!) \sim n \log n$. This is unacceptable if we desire scalability.

To get around this problem, we note that it is not really necessary for players to know the entire permutation. Each player only needs to know its own position and the identities

[2] Note that the above numbered properties for odd and even labeled nodes in the tree are also correct for short and long players in the definitive round.

Algorithm 2 Dealer's Protocol

Parameters: $\mathbb{F}$ is a field of size $q > n$ (to represent messages in the algorithm) n players with distinct identifiers $1, 2, \ldots n \in \mathbb{F}$. $\beta \in (0, 1)$ is geometric distribution parameter. Complete binary tree with n leaves, labeled as described in Sect. 3.1 known to everyone.

1. Choose X, Y, independently from a geometric distribution with parameter β and let $L = X + Y$. Round X is the definitive one. Short players will receive full input for $X - 1$ rounds and partial input for round X. Long players will receive full input for $L - 1$ rounds and partial input for round L. For convenience we will create all the inputs for L rounds, and truncate them appropriately before sending them to the players.
2. for round $r = 1$ set $m_1 = 0$
3. For each round r between 1 and L:
   (a) If $r < L$, then choose a random permutation $\pi_r \in S_n$. Else, randomly choose a permutation π_L which is random subject to the constraint that all the long players (determined by π_X) are assigned to odd labels under π_L.
   (b) *Preparing labeling information:*
   For all players j
      i Set player j's label to $\pi_r(j)$. player j will be assigned to all nodes marked $\pi_r(j)$ in the tree.
      ii Let $\nu_r(j)$ be the set of all neighbors of nodes labeled $\pi_r(j)$ in the tree. $\pi_r(j)$ has a constant number of neighbors.
      iii Let $\pi_r^{-1}(\nu_r(j))$ be the tuple of identities of all the players assigned to a node in set $\nu_r(j)$. Thus, it is the tuple of identities of all neighbors of j.
      iv Generate the labeling information tuple, $(\pi_r(j), \pi_r^{-1}(\nu_r(j)))$.
      v Mask the labeling information tuple, $P_r^j = (\pi_r(j), \pi_r^{-1}(\nu_r(j))) + m_r$.
   (c) *Preparing shares of the secret:*
      i If $r = X$, then $s_r \leftarrow$ true secret.
      Else, $s_r \leftarrow$ random element of $\mathcal{S}$
      ii Create shares of s_r by calling *RecursiveShares* (root, s_r).
   (d) *Preparing mask shares for the next round:*
      i Choose a random mask $m_{r+1} \in \mathbb{F}^c$.
      ii Create shares of m_{r+1} by calling *RecursiveShares* (root, m_{r+1}).
   (e) Create tags and verification functions for all the messages to be sent in round r
   (f) For each player j, j's (full) input I_r^j for round r consists of P_r^j, shares of m_{r+1} and s_r corresponding to node $\pi_r(j)$, tags to authenticate all messages to be sent by j and verification vectors for all the messages to be received by j. Partial input $\tilde{I}_r^j$ consists of all of the above except the authentication tags for sending messages to your children (in the down-stage).
4. Identify the short players as those players j who are at odd numbered nodes in the definitive iteration, *i.e.*, $\pi_X(j)$ is odd.
5. For each short player j, send j the list $I_1^j, \ldots I_{X-1}^j, \tilde{I}_X^j$.
6. For each long player j, send j the list $I_1^j, \ldots I_{L-1}^j, \tilde{I}_L^j$.

Algorithm 3 Protocol for Player j

Parameters: S=0; M=0
If at any time you receive spurious messages (messages not expected under the protocol), ignore them.
On round r:

Up-Stage:

1. $m_r = M$
2. Subtract m_r from P_r^j to find out your position in the tree and the identities of your neighbors for round r.
3. (as a player at a leaf) Send your shares of s_r and m_{r+1} along with their tags to your parent in the tree.
4. (as a player at an internal node)
   (a) Receive (intermediate) shares of s_r and m_{r+1} and tags from left and right chidren. Use the appropriate verification vectors to check that correct messages have been sent. If a fault is detected (missing or incorrect message) output S and quit.
   (b) For each of s_r and m_{r+1}: interpolate them from their shares. This is your share.
   (c) If you are not at the root, send the above reconstructed shares of s_r and m_{r+1} to your parent(s) along with the appropriate tags. If you *are* at the root, these shares are the actual values of s_r and m_{r+1}.

Down-Stage:

1. (as a player at the root) Set $S = s_r$ and $M = m_{r+1}$ and send these values along with authentification tags to your left and right children.
2. (as a player at a non-root node)
   (a) Receive s_r and m_{r+1} and tags from your parent and use verificaton vectors to check them. If fault detected, output S and quit.
   (b) Set $S = s_r$ and $M = m_{r+1}$.
3. (as a player at a non-root internal node) Send s_r and m_{r+1} to your children along with the appropriate tags. If you are a short player and have no authentication tags, output s_r and quit.

$r \leftarrow r + 1$

of its neighbors. We only need a field of size order n to encode this, and so, symbols of this field may be transmitted with messages of length $O(\log n)$. Since it is dificult via share reconstruction to transmit different messages to the leaves of the tree, we simply provide each player with a list of positional data for the entire game. But in order that players do not know their positional data for a round before actually getting to that round, this data is masked by adding in a random element of the field. Positional data for the first round is sent unmasked. The players also receive iterated shares of the masks for the next round. Thus, in each round, players reconstruct a mask, and use it to unmask the positional data and reposition themselves for the next round.

For each round, the full input consists of the following:

- iterated shares of a purported secret (the true secret in the definitive round);
- masked versions of positional data for the current round (position and identities of neighbors in the tree);
- shares of the mask for the *next round* of positional data;
- tags for all the messages to be sent; and
- verification vectors for all the messages to be received.

As mentioned earlier, round X is the definitive round, when the encoded symbol is the true secret. Short players

receive full input for every round prior to this round. For round X they only receive partial input. Long players receive input for $X+Y$ rounds. However, they, too receive only partial input for their last block of input. Otherwise, a player would be able to distinguish whether or not he is a short player by looking at his last block of input. Here, partial input consists of all of the pieces of data from the full input, *except* the tags to send the decoded message to your children in the down stage of the round.

Since in the definitive round the short players (with odd labels) are in the level above the leaves, and all the long players are at internal nodes higher than that in the tree, the long players have learned the secret before the short players, although since they have input for more rounds of the game, they do not know (*i.e.*, cannot guess) that it is the definitive round, and that the secret they have learned is in fact the true secret. Thus, they send the secret down the tree, and eventually it gets to the short players. Thus, the short players learn the secret as well. Since they have no more input they know that the game is over and the secret is the true one. However, since they do not have any more authentication data, they cannot gain by remaining in the game and trying to fool the others into thinking that the secret has not yet been reconstructed. Finally, when the long players do not receive a message at the end of the definitive iteration, they too realize the game has ended and output the correct secret.

3.4 Discussion

A careful reader may ask why it is necessary to permute the players. First, we note that if we never permute the players, then the short and long players know who they are at the beginning of the game. In the game, there is an event with small probability that the short players have an array of size exactly one less than the long players (e.g., short players have array of size one and long players have array of size two). If there are only two rounds remaining in the list of long player, he learns the one to last (the current) round is the definitive round since Y is at least 1. It is crucial for our analysis that long players do not know if the current round is the definitive round. Since, in this case, the long player at the root node can learn the secret without revealing it to any other players.

What if the players are only permuted in the definitive round? Then, when the players see that a permutation occurs, they know they are in the definitive round. Thus, again the long player(s) at the root node will be able to learn the secret without allowing other players to learn it. Hence, we need more than one random permutation during the algorithm. For simplicity of algorithmic presentation and analysis, we permute the players at every round.

A careful reader might wonder why we do not first construct a new scalable protocol for rational broadcast and then use it in the existing protocols for RSS. They are two main reasons. First of all, we note that composing two mechanisms that are Nash equilibria does not necessarily give a new mechanism that is a Nash equilibrium. In general, composability of mechanisms and generating a new one with specific properties is not an easy problem. Second, it seems difficult to come up a mechanism that can broadcast in a scalable manner, because of the linger avoiding assumption we make. In particular, it seems that all players must participate in the broadcast and learn the result of the broadcast at the same time. We achieve scalability by never performing broadcast.

4 Analysis of algorithm for all players present

In this section, we show that the secret sharing scheme we have described is in fact a scalable n-out-of-n secret sharing scheme, and that it is a Nash equilibrium in which all the players learn the secret.

Recall that in Lemma 1 we show our rescursive scheme for encoding a value into n iterated shares (Algorithm 1) is an n-out-of-n scheme.

We will focus our attention on showing that it is a Nash equilibrium for all the players to follow our protocol. Consider player j and suppose that all other players are committed to following the protocol. The next lemma gives a necessary criterion for j to have an incentive to cheat.

Lemma 2 *If all other players are following the protocol, player j prefers to also follow the protocol, unless his probability of successfully cheating is at least* $\frac{U_j - U_j^-}{U_j^+ - U_j^-}$.

Proof Suppose j is considering deviating from the protocol. We will consider the deviation to be successful if either j learns the secret right away, with or without being caught, or he does not get caught and is therefore still in a position to learn it later. The deviation will have failed if it is detected, causing the game to end without j learning the secret. Let p_j be the probability that the deviation succeeds. The maximum utility that j can get is U_j^+. With probability $1 - p_j$, the game ends without j learning the secret, in which case the maximum payoff possible is U_j^-. Thus, a player's expected utility from cheating while everyone else follows the protocol is at most $p_j U_j^+ + (1 - p_j)U_j^-$.

On the other hand, if everyone else follows the protocol, then following the protocol guarantees a utility of at least U_j. Thus, the protocol will be a Nash equilibrium if $U_j > p_j U_j^+ + (1 - p_j)U_j^-$. By rearranging the terms, we have a Nash equilibrium if

$$p_j < \frac{U_j - U_j^-}{U_j^+ - U_j^-}.$$

□

When and how might a player cheat? We note that since players are not required to commit to their strategy before starting the game, and since the progression of the game reveals information, a player may as well defer his decision to cheat in a future round until that future round. Thus, at any given time, the decision facing the player is whether to cheat in the current round. In order to weigh the benefits of such a decision, the player needs an estimate of whether the current round is likely to be the definitive one.

As remarked earlier, the purpose of having short and long players is to create uncertainty about when the definitive round of the game is, until it is too late to gain from this information.

The players know that X is chosen from a geometric distribution with parameter β. Thus, *a priori* the probability that X takes on any particular value is at most β, the most likely being $X = 1$, whose probability is exactly β. As the game progresses, players receive partial information about the value of X; as soon as they receive their inputs they can eliminate all values of X larger than their input length, if the game did not end on the first round, they learn that $X \neq 1$ and so on. Clearly, when a player reaches his last block of input, he knows that the current round is definitive. The next lemma shows that until that stage, a player's estimate that the current round is definitive remains small.

Lemma 3 *Let j be a player who initially received input for $k > 1$ rounds of the game, and let $1 \leq r < k$ be the current round. Then, j's estimate of the probability that the current round is definitive, conditioned on all the information he has learned, is at most 2β.*

Proof Let L_j denote the random variable which is the initial input length of player j. Then, we know that

$$L_j = \begin{cases} X & \text{if } j \text{ is a short player,} \\ X + Y & \text{if } j \text{ is a long player.} \end{cases} \tag{2}$$

Also, let $\mathcal{L}_j$ denote the event that j is a long player and $\mathcal{L}_j^{\mathrm{C}}$ the event that j is a short player. By hypothesis, the current round is $r \geq 1$, and player j received an initial input of length $k > r$. What information does player j know in round r?

- Since his initial input was of length k he knows that $L_j = k$ and $X \leq k$ and moreover, that $X = k$ if and only if $\mathcal{L}_j^{\mathrm{C}}$.
- Since the game has entered round r he knows that $X \geq r$.
- He knows his position $\pi_r(j)$ and the identities of his neighbors in round r.
- He also has learned s_r, m_{r+1} and using the latter to unmask his positional data, he knows $\pi_{r+1}(j)$ and the identities of his neighbors in round $r+1$. Technically, he learns these just prior to his turn in the downstage in round r, but this is fine, as we will argue later that no player ever has any reason to cheat during the upstage of a round.

We note that knowing s_r does not benefit player j in any way as far as estimating the probability that $X = r$ goes, since s_r is equally likely to be any element of $\mathcal{S}$, independently of X. Similarly, knowing the identities of his neighbors does not affect his estimate, since all other players are equally likely to be his neighbors independently of X.

On the other hand, knowing $\pi_r(j)$ and $\pi_{r+i}(j)$ does affect the estimate. By construction:

- In the definitive iteration, short players have odd labels, and long players have even labels; and
- Each player has an odd label in his last round of input.

Thus, if $\pi_r(j)$ is odd, then player j knows that the current round is not definitive, since if $X = r$, then $k > r$ implies that j is a long player and should have an even label. In particular, conditioned on everything he knows, $\Pr(X = r) = 0$. Since $2\beta > 0$ the lemma is proved in this case.

For the remainder of the proof, we will assume that $\pi_r(j)$ is even and denote this event $\mathcal{E}_r$. Consider the case $\pi_{r+1}(j)$. If $k = r+1$, then we know that $\pi_{r+1}(j)$ is odd by construction and knowing this contains no additional information over knowing $L_j = k$. On the other hand, when $k > r+1$, if $\pi_{r+1}(j)$ is odd, then player j knows that X cannot be $r+1$ and this affects the probability that $X = r$.

Let $b \in \{0, 1\}$ be the observed parity of $\pi_{r+1}(j)$ and let $\mathcal{E}_{r+1}^b$ denote the event that the parity of $\pi_{r+1}(j)$ is b. Note that if $k = r+1$ we must have $b = 1$.

Let $P_{j,r}$ be player j's estimate that the current round, r, is definitive, conditioned on everything he knows. Then,

$$\begin{aligned} P_{j,r} &= \Pr(X = r \mid L_j = k \ \wedge\ r \leq X \leq k \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b) \\ &= \frac{\Pr(X = r \ \wedge\ L_j = k \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b)}{\Pr(L_j = k \ \wedge\ t \leq X \leq k \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b)} \\ &\leq \frac{\Pr(X = r \ \wedge\ L_j = k \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b)}{\Pr(L_j = k \ \wedge\ X \in \{r, k\} \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b)}, \end{aligned}$$

where the inequality follows from the fact that the event $X \in \{r, k\}$ is a subset of the event $r \leq X \leq k$.

If the current round *is* definitive *i.e.*, $X = r$, then j is not a short player, and $L_j = X + Y$. So the event $X = r \ \wedge\ L_j = k \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b$ is the same as the event $\mathcal{L}_j \ \wedge\ X = r \ \wedge\ Y = k - r \ \wedge\ \mathcal{E}_{r+1}^b$. Note that $\mathcal{E}_r$ is implied by $\mathcal{L}_j \ \wedge\ X = r$ and can therefore be dropped.

For the denominator, the event $L_j = k \ \wedge\ X \in \{r, k\} \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b$ can be split into the union of disjoint events $\mathcal{L}_j^{\mathrm{C}} \ \wedge\ X = k \ \wedge\ \mathcal{E}_r \ \wedge\ \mathcal{E}_{r+1}^b$ and $\mathcal{L}_j \ \wedge\ X = r \ \wedge\ Y = k - r \ \wedge\ \mathcal{E}_{r+1}^b$. The latter summand is the same as the numerator, and loses

the $\mathcal{E}_r$ term for the same reason. Making these substitutions in the above expression, we get

$$P_{j,r} \leq A/B,$$

where

$$A = \Pr(\mathcal{L}_j \ \wedge \ X = r \ \wedge \ Y = k - r \ \wedge \ \mathcal{E}^b_{r+1})$$

and

$$\begin{aligned} B = {} & \Pr(\mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r \ \wedge \ \mathcal{E}^b_{r+1}) \\ & + \Pr(\mathcal{L}_j \ \wedge \ X = t \ \wedge \ Y = k - t \ \wedge \ \mathcal{E}^b_{r+1}). \end{aligned}$$

The random variables X, Y and the indicator that j is a long player are independent. Thus the numerator of the above expression becomes

$$\begin{aligned} & \Pr(\mathcal{L}_j \ \wedge \ X = r \ \wedge \ Y = k - r \ \wedge \ \mathcal{E}^b_{r+1}) \\ & \quad = \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}_j \ \wedge \ X = r \ \wedge \ Y = k - r) \\ & \qquad \times \Pr(\mathcal{L}_j) \Pr(X = r) \Pr(Y = k - r) \\ & \quad = \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}_j \ \wedge \ X = r \ \wedge \ Y = k - r) \\ & \qquad \times \frac{\lfloor n/2 \rfloor}{n} (1 - \beta)^{r-1} \beta (1 - \beta)^{k-r-1} \beta \\ & \quad = \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}_j \ \wedge \ X = r \ \wedge \ Y = k - r) \\ & \qquad \times \frac{\lfloor n/2 \rfloor}{n} (1 - \beta)^{k-2} \beta^2. \end{aligned}$$

Similarly we tackle the first term in the denominator. Since X and all the π_i are independent and $k \neq r$, then π_r and π_X are independent conditioned on $X = k$. It follows that the events $\mathcal{L}^{\mathrm{C}}_j = \pi_X(j)$ is odd; $\mathcal{E}_r = \pi_r(j)$ is even; and $X = r$ are independent. Thus, we have the following.

$$\begin{aligned} & \Pr(\mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r \ \wedge \ \mathcal{E}^b_{r+1}) \\ & \quad = \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r) \\ & \qquad \times \Pr(\mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r) \\ & \quad = \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r) \\ & \qquad \times \Pr(\mathcal{L}^{\mathrm{C}}_j) \Pr(\mathcal{E}_r) \Pr(X = k) \\ & \quad = \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r) \\ & \qquad \times \frac{\lceil n/2 \rceil \lfloor n/2 \rfloor}{n^2} (1 - \beta)^{k-1} \beta \\ & \quad \geq \Pr(\mathcal{E}^b_{r+1} | \mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r) \\ & \qquad \times \frac{\lfloor n/2 \rfloor}{2n} (1 - \beta)^{k-1} \beta. \end{aligned}$$

If $k > r+1$ then π_X and π_{r+1} are independent conditioned on X being either r or k, and hence, the events $\mathcal{E}^b_{r+1}$ and $\mathcal{L}_j \wedge X = r \wedge Y = k - r$ are independent, as are the events $\mathcal{E}^b_{r+1}$ and $\mathcal{L}^{\mathrm{C}}_j \wedge X = k \wedge \mathcal{E}_r$. It follows that $\Pr(\mathcal{E}^b_{r+1} | \mathcal{L}_j \wedge X = r \wedge Y = k - r)$ and $\Pr(\mathcal{E}^b_{r+1} | \mathcal{L}^{\mathrm{C}}_j \wedge X = k \wedge \mathcal{E}_r)$ both equal $\Pr(\mathcal{E}^b_{r+1})$. On the other hand if $k = r + 1$ then $b = 1$ and $\mathcal{E}^b_{r+1}$ is implied in both cases Thus, $\Pr(\mathcal{E}^b_{r+1} | \mathcal{L}_j \ \wedge \ X = r \ \wedge \ Y = k - r)$ and $\Pr(\mathcal{E}^b_{r+1} | \mathcal{L}^{\mathrm{C}}_j \ \wedge \ X = k \ \wedge \ \mathcal{E}_r)$ are both 1. Either way, they are equal, and since their common value occurs in the numerator as well as in both terms in the denominator, it simply cancels out.

Putting everything together we see that

$$\begin{aligned} P_{j,r} & \leq \frac{\frac{\lfloor n/2 \rfloor}{n} (1 - \beta)^{k-2} \beta^2}{\frac{\lfloor n/2 \rfloor}{2n} (1 - \beta)^{k-1} \beta + \frac{\lfloor n/2 \rfloor}{n} (1 - \beta)^{k-2} \beta^2} \\ & = \frac{\beta}{\frac{1}{2}(1 - \beta) + \beta} \\ & \leq 2\beta. \end{aligned}$$

□

We remark that although in the above proof we have bounded player j's estimate during the down-stage, a nearly identical proof shows the same bound for the up-stage (when $\pi_{r+1}(j)$ is unknown). We are now ready to prove the main theorem.

Proof of Theorem 1 We will begin by showing that the protocol is a Nash equlibrium in which all the players learn the secret.

Suppose all the players follow the protocol. In every round, during the up-stage, players send their shares up toward the root where they are decoded. During the down-stage, the reconstructed secret and mask are sent back toward the leaves. If the odd players in the round do not drop out at the end of the round, then the game continues into the next round. In the definitive round, the real secret is reconstructed at the root and all the even labeled players, who are long players learn it first. Once it gets to the short players with odd labels, they drop out of the game since they have no tags to send any more messages. This signals the end of the game to the long players who then realize that the current reconstructed secret is the true one. Thus if all the players follow the protocol, everyone learns the secret.

Suppose all players other than j are following the protocol. We want to show that player j prefers following the protocol over deviating.

At the beginning of the game, each player has set their current guess for the secret to 0. If no cheating occurred before the current round r then during round $r - 1$ all the players set their current guess to s_{r-1}. Thus, at the beginning of round r, all players have the same guess for the secret. (Round $r - 1$ has been eliminated as the definitive one, but s_{r-1} still has probability $1/|\mathcal{S}|$ of being the true secret). Moreover, since by Lemma 1 partial information about the shares reveals no information about the symbol they encode, throughout the up-stage player j has no better guess than s_{r-1} for the secret. Since $\mathcal{U} < |\mathcal{S}|$, it is strictly better for j not to leave the game in the up-stage. If j sends incorrect messages in the

up-stage, then even if he is not caught(which results in not learning the secret), this deviation will cause an incorrect value to be decoded instead of s_r. This results in j not learning the secret if r happened to be the definitive round. Thus, j has no incentive to deviate in the up-stage.

Now, what about the down-stage? If j is on his last round of input, then he is a short player, and knows that the current round is definitive. At the same time, this means that he is an odd-labeled player and by the construction of the communication tree he cannot prevent anyone from learning the secret. Moreover, even if he successfully fakes a tag in order to convince the unique long player below him that the game has not ended, that player will detect in the next round that all other players have left the game and will therefore still output the correct secret. Thus this deviation does not change the outcome of the game, namely that all players learn the secret. It follows that j does not gain anything by this deviation (although he also does not lose anything by it). Effectively, if j is a short player on his last round of input, it is too late for him to improve his payoff by deviating.

If j is not on his last round of the input and is at an odd labeled node, then, as remarked in the proof of Lemma 3, he *knows* the current round is not definitive. So, cheating would be equivalent to randomly guessing the secret that is correct with probability only $1/|S|$. This is worse than following the protocol by Eq. (1).

Now, suppose that the current round, r, is not j's last round of the input and j is at an even labeled node. Note that any spurious messages sent by player j to players that are not expecting them, will be ignored. Also, any action involving not sending a message that is expected will be detected immediately, only by the involved players at first, but the knowledge will quickly propagate to all the players, before the end of the up-stage of the next round. Since detection of deviation causes other players to quit immediately, effectively such actions amount to player j leaving the game. Thus, the possible deviations we need to analyze for player j are:

- Leave the game with or without sending fake messages first, and;
- Send fake messages to one or both children and hope to stay in the game by not being caught.

Let $P_{j,r}$ be j's estimate of the probability that the current round is definitive. Then, by Lemma 3, $P_{j,r} \leq 2\beta$.

If player j leaves the game and outputs the value s_r then the probability that he has output the right value is $P_{j,r} + (1 - P_{j,r})/|\mathcal{S}|$, and since he has left the game, he has no later opportunity to improve that probability. By Lemma 2, in order to discourage this deviation it is sufficient if

$$P_{j,r} + \frac{(1 - P_{j,r})}{|\mathcal{S}|} \leq \frac{U_j - U_j^-}{U_j^+ - U_j^-}. \tag{3}$$

Now consider the other kind of deviation. Suppose instead of sending s_r to his descendents, player j sends a fake value to one or both of them. Let α be the probability that he is not caught. By Proposition 1 we know that $\alpha = \frac{1}{q-1}$ if he sends a fake message to only one of his children, and, since the two verification functions are chosen independently by the dealer, $\alpha = \frac{1}{(q-1)^2}$ if he fakes both messages. If he is not caught, and if the current round is definitive, then player j has learned the true secret and has prevented some of his descendants from learning it. If the current round was not definitive and his deviation was not detected, the game continues and since the values s_i are all independent, it does not affect the next round.[3] This means that player j can either revert to following the protocol and guarantee learning the secret along with everyone else, or he may find further opportunities to cheat.

On the other hand, if the faked message is detected, which happens with probability $1 - \alpha$, then the game ends right away and player j outputs s_r. In this case, there is still a $P_{j,r}$ chance that the current round was definitive and an additional $(1 - P_{j,r})/|\mathcal{S}|$ chance that the value s_r was correct despite the current round not having been definitive. So the probability that the deviation succeeds is

$$\alpha + (1 - \alpha)\left(P_{j,r} + \frac{(1 - P_{j,r})}{|\mathcal{S}|}\right).$$

As this quantity is bigger when $\alpha = \frac{1}{q-1}$, faking only one message dominates faking both messages. Thus, again by Lemma 2, to discourage this deviation, it is sufficient if

$$\frac{1}{q-1} + \frac{q-2}{q-1}\left(P_{j,r} + \frac{(1 - P_{j,r})}{|\mathcal{S}|}\right) \leq \frac{U_j - U_j^-}{U_j^+ - U_j^-}. \tag{4}$$

Moreover, note that (4) implies (3). Thus for a Nash equilibrium, it is sufficient to show (4).

For the remainder of the proof, we are going to assume that n is sufficiently large, specifically that $n \geq \frac{2\mathcal{U}|\mathcal{S}|}{|\mathcal{S}|-\mathcal{U}}$. We will discuss the modifications required when $3 \leq n < \frac{2\mathcal{U}|\mathcal{S}|}{|\mathcal{S}|-\mathcal{U}}$ in Sect. 4.1.1.

We have so far not specified β. We do this now. Let

$$\beta = \frac{|\mathcal{S}| - \mathcal{U}}{4\mathcal{U}|\mathcal{S}|}.$$

Then, $1/\beta = O(1)$ so that the expected number of rounds in the game is constant.

[3] This is why player j does not try to fake the mask m_{r+1} - a successfully transmitted incorrect m_{r+1} will wreak havoc in the next round, since some players will be talking to the wrong players.

To show (4), recall that $q > n$, so that $q - 1 \geq n$. We have

$$\begin{aligned}
&\frac{1}{q-1} + \frac{q-2}{q-1}\left(P_{j,r} + \frac{(1-P_{j,r})}{|\mathcal{S}|}\right) \\
&\quad < \frac{1}{n} + \left(P_{j,r} + \frac{(1-P_{j,r})}{|\mathcal{S}|}\right) \\
&\quad = \frac{1}{n} + \frac{|\mathcal{S}|-1}{|\mathcal{S}|}P_{j,r} + \frac{1}{|\mathcal{S}|} \\
&\quad \leq \frac{|\mathcal{S}|-\mathcal{U}}{2\mathcal{U}|\mathcal{S}|} + 2\beta + \frac{1}{|\mathcal{S}|} \\
&\quad \leq \frac{|\mathcal{S}|-\mathcal{U}}{2\mathcal{U}|\mathcal{S}|} + 2\frac{|\mathcal{S}|-\mathcal{U}}{4\mathcal{U}|\mathcal{S}|} + \frac{1}{|\mathcal{S}|} \\
&\quad = \frac{|\mathcal{S}|-\mathcal{U}}{\mathcal{U}|\mathcal{S}|} + \frac{1}{|\mathcal{S}|} \\
&\quad = \frac{1}{\mathcal{U}}
\end{aligned}$$

as desired.

Finally, we analyze the resource costs of our protocol. The communication tree has $2n - 1$ nodes. In each round, each player is mapped to one leaf and one internal node. Players only communicate with their neighbors in the tree. So on each round, during the up-stage each player sends up to three messages: one to his parent when he is a leaf; one to his parent when he is a non-root internal node; and if he is a child of the root and n is even, he has to send an additional message because there are two players at the root.

During the down stage, each player sends two messages, to his two children. Thus, each player sends at most five messages per round. Each message consists of four elements of $\mathbb{F}$ (shares of s_r, m_{r+1} and two tags) each of which is represented as $O(\log n)$, since $n < q \leq 2n$. Thus, each player send $O(\log n)$ bits per round. Since the expected number of rounds is $1/\beta$, which is constant, each player sends only $O(\log n)$ bits during the course of the game. Finally, since the tree has depth $O(\log n)$, the number of rounds is constant (in expectation). Note that the communication is synchronous, which follows that the expected latency is $O(\log n)$.

4.1 Some remarks

4.1.1 The case of a small number of players

When the number of players is a constant greater than 2, then scalability is not an issue, and one might hope to simply use the algorithm of Kol and Naor [10] by simulating non-simultaneous broadcast channels with secure private channels. Unfortunately their algorithm only provides an ϵ-Nash equilibrium, since the unique short player has a small chance of successfully pretending the game has not ended.

In our algorithm, $\lceil n/2 \rceil$ players are short players. In particular, even for $n = 3$, there are at least two short players, and none of the short players can increase their expected payoff by cheating alone. Thus, we obtain a Nash equilibrium, provided that we can prove inequality (4). The above proof used the fact that n was at least $\frac{2\mathcal{U}|\mathcal{S}|}{|\mathcal{S}|-\mathcal{U}}$, so we need a separate argument.

However, scalability is immediate when n is constant. Thus, we have more leeway to choose a larger field to work with.[4] So, we can work in a prime field of size q where

$$\max\{|\mathcal{S}|, \frac{2\mathcal{U}|\mathcal{S}|}{|\mathcal{S}|-\mathcal{U}}\} < q \leq 2\max\left\{|\mathcal{S}|, \frac{2\mathcal{U}|\mathcal{S}|}{|\mathcal{S}|-\mathcal{U}}\right\}$$

and the proof of (4) goes through as before, giving us a Nash equilibrium.

4.1.2 Nash equilibria versus strict nash equilibria

An n-tuple of strategies is called a strict Nash equilibrium if when all other players are following the prescribed strategy, a player unilaterally deviating achieves a strictly worse expected payoff than he would by following the equilibrium strategy.

Our algorithm fails to be a strict Nash equilibrium, for the following reasons:

- Any player may, at any time, send spurious messages that are not part of the protocol, to players that are not his neighbors in the tree. Such messages will be ignored by their recipients, who are following the protocol.
- At the end of the definitive round, a short player may try to fake a tag and send a message to the long player below him. This may go undetected with some small probability, but as noted in the proof, even in this case, it cannot fool that long player into outputting the wrong secret.

Our proof shows that our algorithm *does* have the property that any player deviating from the protocol in one of the above ways does not increase his payoff, and moreover *does not affect any other player's payoff either.* In other words, if a player deviating unilaterally from our protocol, does so in a manner that changes some other player's payoff, then he strictly reduces his own expected payoff. In this weaker sense, the equilibrium is strict.

4.1.3 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.

[4] The upper bound of $O(n)$ on the field size came from the desire to keep $4 \log q$, which is the size of an individual message, small.

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.

As in [10], we protect against backwards induction by having both long and short players. As the above analysis shows, we can ensure 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 his lists, he 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.

The backwards induction problem may occur with protocols that make cryptographic assumptions. If there is a round, ℓ, in which enough time has passed, so that even a computationally bounded player can break the cryptography, then it is globally known that the protocol will end at round ℓ. Thus, even though ℓ may be far off in the future, by backwards induction, there is no incentive for a player to follow the protocol even at the beginning. We note that in out t-out-of-n protocol, there is no such round since the keys are refreshed every round. Moreover, the information about the key for each round is shared until that round and nobody knows it beforehand.

5 Algorithm for t-out-of-n secret sharing

In this section, we discuss t-out-of-n secret sharing where $t < n$. Here, we want any subset of t or more of the players to be able to reconstruct the secret. However, fewer than t players should *not* be able to reconstruct the secret on their own.

We will assume, as in previous work, that the set of active players is known to all the active players, before the start of the players' protocol. However, this set is not known to the dealer at the time of share creation.

Our algorithm will be a hybrid of the Kol and Noar [10] scheme for t out of n players and our scheme described above for n out of n players. As in [10], we now have a single short player and the secret shares now include an indicator bit to enable any subset of t players to reconstruct the secret, even if that subset does not contain the short player. The existence of the short player is necessary to avoid backwards induction. See the work of Kol and Naor [10] for details.

Since the algorithm is a variant of the n-out-of-n scheme, for conciseness, we briefly describe the protocol focusing only on the places where the two algorithms differ. We first describe sharing and reconstruction (Sects. 5.1 and 5.2) without authentication, and then show how to modify to ensure authentication in Sect. 5.3.

5.1 Creating the shares

We assume each player has an id that is known to all other players. Moreover, each player initially has a label from 2 to $n+1$ based on their sorted ids. These labels may change during the course of the protocol. In particular, in each round, exactly one player will have its label change to 1, and the player player previously labeled 1 will revert to its initial label. At the start, the dealer selects a random player to be labeled 1 in the first round and sends its id to all players.

To ensure reconstructability, in addition to the secret, the shares will need to encode some more information. Specifically, they will also encode an indicator bit to identify the definitive round, and they will also encode the label of the player that will be labeled 1 for the following round. For a given round r, we let s_r be the potential secret, b_r be the indicator bit, and ℓ_r be the label of the player who will have label 1 in round $r+1$.

The dealer samples values X and Y independently from a geometric distribution with parameter β and lets $L = X + Y$. The game will have L rounds, and round X will be the definitive one.

The player labeled 1 in the definitive round is *the short player*.

For each round $1 \le r \le L$, the dealer proceeds as follows. The value ℓ_r is set to a number selected uniformly at random between 2 and $n+1$. Then, if $r = X$, s_r is set to be the true secret, $b_r \leftarrow 1$. Otherwise if $r \neq X$, s_r is set to a random value in the set $\mathcal{S}$ and $b_r \leftarrow 0$.

The dealer prepares the players' inputs as follows. For rounds, $1 \le r \le L$, the dealer uses Shamir's (t, n)-secret sharing to generate n different shares of the tuple $\langle r, s_r, b_r, \ell_r \rangle$ and also the hash and tags for each message as described in Sect. 5.3. For rounds $1 \le r \le X$, the dealer sends out shares to all the players. For rounds $X+1 \le r \le L$, the dealer sends out shares to all players *except* the short player.

5.2 Reconstruction phase

For the reconstruction phase, we will make use of the following property of the t-out-of-n Shamir's secret sharing: For every set of shares $s_1, \ldots, s_{t'}$ of cardinality $t' \ge t$, there exist coefficients (Lagrange coefficients) $\lambda_1, \ldots, \lambda_{t'}$, which depend only on the identities of the active players, such that $s = \sum_i^{t'} \lambda_i s_i$. See [9,13] for details.

Let T be the group of t players who are active, and let p_i be the i-th player in T in the sorted order based on their labels. Every active player knows the id and label of all players in T, and the id of the player labeled 1 for the first round. In each round, players generate a complete binary tree with t leaves and assign player p_i to the i-th leaf node, going from left to right. For all $1 < i \le t$, player p_i is also assigned to

an internal node of the tree from root node, going from top to bottom and from left to right.

Each player i at a leaf node multiplies his shares of $\langle r, s_r, b_r, \ell_r \rangle$ by λ_i and sends the result to his parent in the tree. Players at internal nodes of the tree add the received shares together and send the result up. The root node learns the tuple $\langle r, s_r, b_r, \ell_r \rangle$, and sends it down in the tree. Subsequently, each player sends the result down until everybody learns it. If a player does not receive the correct messages from his children or parent, or if he receives a message with indicator 1, he outputs the last secret that was associated with an indicator 0.

5.3 MAC scheme for t-out-of-n

In the t-out-of-n setting, the set of active players, and hence, the position of the players in the reconstruction tree are unknown to the dealer. Thus, unlike the n-out-of-n case, the dealer cannot generate a tag for the sender and a unique matching verification vector for the potential receiver of the message. Instead, the dealer uses the fact that the same message is propagated in the tree for the down stage. Thus, he uses digital signature to sign messages before sharing them.

Moreover, as in the n-out-of-n case, no player has an incentive to cheat in the up stage of a round, since that results in nobody recovering the secret, including, in particular, the cheater. Thus there is no need to have a verification scheme for the messages in the up stage.

We now formally describe this scheme. For each round r of the protocol, the dealer generates a pair of keys for a digital signature scheme, a secret key SK_r and a public key PK_r. The public key PK_1 is sent to all the players as the first part of their input share. For round $r \geq 1$, after constructing the tuple $\langle r, s_r, b_r, \ell_r \rangle$, the dealer appends it with the private key PK_{r+1}. He then signs the resulting message $\langle r, s_r, b_r, \ell_r, PK_{r+1} \rangle$ using his private key SK_r, and then creates Shamir shares of the resulting *signed* message. These shares are then sent to the players. During the reconstruction phase, the player at the root node recovers the *signed* tuple $\langle r, s_r, b_r, \ell_r, PK_{r+1} \rangle$ and uses the dealer's public key PK_r to verify that it was indeed generated by the dealer, and not forged. If it passes the verification, he forwards the *signed* tuple to his children. Otherwise, he quits the game and outputs the last secret that was associated with an indicator zero. This process of checking and forwarding is repeated by all internal nodes in the tree until all the players have learned the tuple $\langle r, s_r, b_r, \ell_r, PK_{r+1} \rangle$ for round r. Thus, each key pair is used to sign only one message. Based on the security of digital signatures, the probability that a player who does not know the dealer's secret key, can successfully forge a message and make it look like it came from the dealer is negligible. Thus this scheme gives us cryptographic security for our algorithm. We note that, unlike in the n-out-of-n case, we do *not* achieve information theoretic security.

5.4 Analysis

In this section, we will prove Theorem 2.

Lemma 4 *Let j be any player who initially received input for $k > 1$ rounds of the game, and let r be the current round $1 \leq r < k$ such that $r \neq X + 1$. Then, j's estimate of the probability that round r is definitive, conditioned on all the information he has learned, is at most 2β.*

Proof The proof is similar to the proof of the n-out-of-n case. Let L_j denote the random variable which is the initial input length of player j. Then, we know that

$$L_j = \begin{cases} X & \text{if } j \text{ is a short player,} \\ X + Y & \text{if } j \text{ is a long player.} \end{cases} \tag{5}$$

Also, let $\mathcal{L}_j$ denote the event that j is a long player and $\mathcal{L}_j^{\mathrm{C}}$ the event that j is a short player. By hypothesis, the current round is $r \geq 1$, and player j received an initial input of length $k > r$. What information does player j know in round r? He knows the following.

- Since his initial input was of length k he knows that $L_j = k$ and $X \leq k$ and moreover, that $X = k$ if and only if $\mathcal{L}_j^{\mathrm{C}}$.
- Since the game has entered round r he knows that $X \geq r - 1$.
- He knows his position and the identities of his neighbors in the tree.
- He also has learned s_{r-1}, s_r, b_{r-1}, b_r. Technically, he learns these just prior to his turn in the downstage in round r, but this is fine, as we argued before, no player ever has any reason to cheat during the upstage of a round.

We note that knowing s_{r-1}, s_r, does not benefit player j in any way as far as estimating the probability that $X = r$ goes, since both s_{r-1}, s_r, are equally likely to be any element of $\mathcal{S}$, independently of X. Similarly, knowing the identities of his neighbors does not affect his estimate, since positions of the players are independent of X.

On the other hand, knowing his label does not affect player j's chance of successful cheating. By construction in the definitive round, the short player has label 1, and long players have other labels larger than 1. However, in the definitive round, the short player already reached its last input, and he knows that this round is the definitive round (without the help of knowing its label).

Moreover, each player thinks that he can have label 1 in his last round of inputs and long players never play until the last round to figure out it is not the case for them. Thus, if

the label is 1 but $k > r$, then player j knows that the current round is not definitive, since if $X = r$, then $k > r$ implies that j is a long player and should not have label 1 in definitive round. In particular, conditioned on everything he knows, $\Pr(X = r) = 0$. Since $2\beta > 0$ the lemma is proved in this case.

Similarly, knowing that the values of b_{r-1}, and b_r are both zero (since $r \neq X + 1$) also does not affect the estimate since that just indicates that the game has not reached the definitive round yet. The remainder of the proof is exactly like the proof for n-out-of-n case (see Lemma 3) and we do not repeat it here. □

Now we can complete the proof for Theorem 2, which we recall here.

Theorem 3 *Let $n \geq 3$ and $t \leq n$. Assuming the players are computationally bounded to polynomial-time algorithms, there exists a protocol for rational t-out-of-n secret sharing with the following properties.*

- *The protocol is an everlasting ϵ-Nash equilibrium in which all active players learn the secret, where ϵ can be any arbitrary positive value.*
- *In expectation, the protocol requires each player to send $O(\log n)$ bits, and has latency $O(\log n)$.*
- *The protocol is not robust to coalition of any size.*

Proof In this proof, we use case analysis for different values of r ($r < X$, $r = X$ and $r = X + 1$) and different kinds of players (short and long).

- When $r < X$, both short and long players have no incentive to deviate the protocol based on Lemma 4.
- When $r = X$ and player j is the short player, then in round r player j is in his last round of his input. Hence, player j knows he is in definitive round. However, he is not willing to cheat in the game since he wants to find the secret of this round and he will be the last one who learns it.
- When $r = X$ and player j is a long player, he has no incentive to cheat in this round based on Lemma 4.
- When $r = X+1$ and player j is a short player, he does not have enough information to participate in the this round, and the only way he can cheat is to successfully forge the messages for the game.
- When $r = X + 1$ and player j is a long player who learns the last round was definitive based on b_r sooner than others since he is in the root, the only way he can cheat in this round is to successfully forge the messages for the game.

Therefore, the only way a player can cheat is to successfully forge a message by breaking the digital signature scheme. Player j gains the largest utility if he cheats in the definitive round after he learns the secret. The probability that a player can successfully break the digital signature scheme is at most $\frac{1}{2^\kappa}$ where κ is the security parameter for digital signature. Thus, if player j decides to cheat and forge a message at the definitive round, with probability $1/2^\kappa$ he can get at most the maximum utility of U_j^+ and with probability of $1 - 1/2^\kappa$ can get at most the maximum utility of $\mathcal{U}_j$. The player who decides to cheat before the definitive round gets the utility of $Uminus_j$ which is less than U_j. Thus, player j's utility is at most $\frac{U_j^+ + (1-2^\kappa)U_j}{2^\kappa}$. On the other hand, if the protocol runs correctly, he has utility at least U_j. Therefore, our protocol is an ϵ-Nash where ϵ is

$$\frac{U_j^+ + (1 - 2^\kappa)U_j}{2^\kappa} - U_j = \frac{U_j^+ + (1 - 2^{\kappa+1})U_j}{2^\kappa}.$$

Solving this equation for κ, we have $\kappa = \max_j \frac{U_j^+ + U_j}{2U_j + \epsilon}$. Finally, similar to [10], since player j gains the largest utility if he cheats in the last round, our protocol is an everlasting ϵ-Nash equilibrium. □

6 Conclusion

We have presented *scalable* mechanisms for rational secret sharing problems. Our algorithms are scalable in the sense that the number of bits sent by each player is $O(\log n)$ and the latency is at most logarithmic in the number of players. For n-out-of-n rational secret sharing, we give a scalable algorithm that is a Nash equilibrium to solve this problem. For t-out-of-n rational secret sharing where, we give a scalable algorithm that is a ϵ-Nash equilibrium.

Several open problems remain. First, while our algorithms lead to a $\Theta(n)$ multiplicative reduction in communication costs for rational secure multiparty computation (MPC), the overall bandwidth for this problem is still very high. We ask: can we design scalable algorithms for rational MPC? This is related to our second open problem which is: can we design scalable algorithms for simulating a class of well-motivated mediators? A final important problem is: can we design coalition-resistant scalable algorithms for rational secret sharing?

Acknowledgments We are grateful to Tom Hayes, Mahdi Zamani, Jonathan Katz, and the anonymous reviewers for their useful discussions and comments.

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: Proceedings of the 25th

Annual ACM Symposium on Principles of Distributed Computing, pp. 53–62. ACM, New York (2006)
2. Asharov, G., Lindell, Y.: Utility dependence in correct and fair rational secret sharing. In: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '09, pp. 559–576. Springer, Berlin (2009). doi:10.1007/978-3-642-03356-8_33. URL http://dx.doi.org/10.1007/978-3-642-03356-8_33
3. Dani, V., Movahedi, M., Rodriguez, Y., Saia, J.: Scalable rational secret sharing. In: Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing. ACM, New York (2011)
4. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient rational secret sharing in standard communication networks. In: Proceedings of the 7th International Conference on Theory of Cryptography, TCC'10, pp. 419–436. Springer, Berlin, (2010). doi:10.1007/978-3-642-11799-2_25. URL http://dx.doi.org/10.1007/978-3-642-11799-2_25
5. Geambasu, R., Kohno, T., Levy, A., Levy, H.: Vanish: Increasing data privacy with self-destructing data. In: Proceedings of the 18th Conference on USENIX Security Symposium, pp. 299–316. USENIX Association, Berkeley (2009)
6. Gordon, S., Katz, J.: Rational secret sharing, revisited. Security and Cryptography for Networks pp. 229–241 (2006)
7. Halpern, J., Teague, V.: Rational secret sharing and multiparty computation: extended abstract. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, p. 632. ACM, New York (2004)
8. King, V., Saia, J.: 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, pp. 420–429. ACM, New York (2010)
9. Knuth, D.E.: The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms. Addison-Wesley Longman Publishing Co., Inc, Boston, MA (1997)
10. Kol, G., Naor, M.: Games for exchanging information. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 423–432. ACM, New York (2008)
11. Lysyanskaya, A., Triandopoulos, N.: Rationality and adversarial behavior in multi-party computation. Adv. Cryptol. CRYPTO **2006**, 180–197 (2006)
12. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 73–85. ACM, New york (1989)
13. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
14. Wegman, M., Carter, J.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. **22**(3), 265–279 (1981)
15. Zhang, Y., Tartary, C., Wang, H.: An efficient rational secret sharing scheme based on the chinese remainder theorem. In: Parampalli, U., Hawkes, P. (eds.) Information Security and Privacy, Lecture Notes in Computer Science, vol. 6812, pp. 259–275. Springer, Berlin (2011). doi:10.1007/978-3-642-22497-3_17. URL http://dx.doi.org/10.1007/978-3-642-22497-3_17