Fair and Sound Secret Sharing from Homomorphic Time-Lock Puzzles

Jodie Knapp and Elizabeth A. Quaglia

Information Security Group, Royal Holloway, University of London,
jodie.knapp.2018@rhul.ac.uk, elizabeth.quaglia@rhul.ac.uk

Abstract. Achieving fairness and soundness in non-simultaneous rational secret sharing schemes has proved to be challenging. On the one hand, soundness can be ensured by providing side information related to the secret as a check, but on the other, this can be used by deviant players to compromise fairness. To overcome this, the idea of incorporating a time delay was suggested in the literature: in particular, time-delay encryption based on memory-bound functions has been put forth as a solution. In this paper, we propose a different approach to achieve such delay, namely using homomorphic time-lock puzzles (HTLPs), introduced at CRYPTO 2019, and construct a fair and sound rational secret sharing scheme in the non-simultaneous setting from HTLPs.
HTLPs are used to embed sub-shares of the secret for a predetermined time. This allows to restore fairness of the secret reconstruction phase, despite players having access to information related to the secret which is required to ensure the soundness of the scheme. Key to our construction is the fact that the time-lock puzzles are homomorphic so that players can compactly evaluate sub-shares. Without this efficiency improvement, players would have to independently solve each puzzle sent from the other players to obtain a share of the secret, which would be computationally inefficient. We argue that achieving both fairness and soundness in a non-simultaneous scheme using a time delay based on CPU-bound functions rather than memory-bound functions is more cost-effective and realistic in relation to the implementation of the construction.

1 Introduction

Threshold secret sharing (SS) schemes provide a way to split a secret into shares such that the secret can be reconstructed by a threshold number of mutually distrustful parties. Knowledge of fewer than the threshold number of shares reveals nothing about the secret [5,37]. SS schemes are an important primitive used in a variety of settings from multiparty computation [7,9], to attribute-based encryption [20,41], and threshold cryptography [4,12]. In a SS scheme, a trusted dealer splits the secret into shares and distributes one to each authorised party. Parties then communicate and process their collective shares in a reconstruction phase. During the communication phase, parties broadcast their shares in one of two ways: *simultaneously* or *non-simultaneously*. That is, with or without synchronicity. Properties of SS schemes are better understood and easier to guarantee in the

simultaneous setting [10], due to the fact that a non-simultaneous construction needs to ensure the final party to communicate is still incentivised to follow the protocol. However, simultaneous schemes are difficult to implement in practice, therefore attention has recently turned to non-simultaneous communication [2].

Typically, in the non-simultaneous setting [17,25,26], schemes consists of rounds, where one round of the reconstruction phase simply translates to a capped period of time in which parties have the opportunity to communicate their share. Parties learn the secret is reconstructed when they reconstruct some publicly known value (for example, an indicator), in what is known as a revelation round [29,22]. The previous round to the revelation round is assumed to be the one in which the secret can be reconstructed from, allowing parties to identify when they will reconstruct the correct secret.

There is abundant literature for cryptographic [3,4,8,18,22,27,28,29,34,38] and game-theoretical SS schemes [2,10,17,19,21,30], two somewhat independent research areas considering honest/malicious parties and rational players, respectively. We refer to Appendix G of [24] for a brief summary of past works. Rational secret sharing (RSS) was introduced by [21], where they consider the problem of secret sharing and multiparty computation assuming players prefer to learn the secret over not learning it, and secondly, prefer that as few as possible other players learn the secret. While for some applications the cryptographic setting is appropriate, for other applications of secret sharing it may be more suitable to view all parties as rational players. RSS is a good approach to capture more interesting scenarios, such as how to motivate or force players to participate honestly and even how a scheme can penalise players for deviant play. Furthermore, modelling players as rational is not limited to assuming players always want to learn the secret above all else. Indeed, as we will explore, an emerging scenario in RSS considers players that prefer to mislead others above learning the secret. For these reasons, our attention focuses on RSS schemes.

In RSS schemes, the outcome of the game influences the players' strategies, as they seek to maximise their payoff. Security of the game requires the strategies of players to be in some form of equilibrium which motivates them to honestly communicate[1]. Achieving an equilibrium between players' strategies is the most natural way to demonstrate a fundamental property of SS schemes, called *fairness* [14,23].

A fair scheme ensures that if a player deviates, the probability that they can recover the shared secret over honest players is negligible. That is, a player is at no advantage in learning the secret if they withhold or dishonestly send a share. In the simultaneous setting, [19,21] both achieve fairness using some form of publicly known indicator and by demonstrating that their protocol is in a form of Nash equilibrium [33]. In the non-simultaneous setting, however, a basic threat to fairness arises: in a (t, n) threshold RSS scheme, the last player out of t can decide not to communicate their share and use all the other players' shares to reconstruct the secret, leaving the $(t-1)$ honest players with an insufficient

[1] See Appendices D.1, and D.2 for further discussion on payoff functions and equilibrium concepts.

number of shares to do so. The rational behaviour of all parties would therefore be to withhold their share. In works such as [17,26,30], fairness can be achieved similarly to the simultaneous setting, whereby players can recognise the revelation round using (or reconstructing) some form of public indicator. However, this only works under the assumption that players prefer everyone to obtain the correct output over misleading others [2,10]. If this assumption does not hold, an alternative way of providing fairness needs to be used, as another property of SS schemes is no longer ensured, *soundness*.

In RSS schemes, *soundness* [2,10] ensures players never reconstruct an incorrect secret except with negligible probability. In other words, honest players are guaranteed to output a correct value, or a special abort symbol $\perp$ [2]. Soundness is becoming of emerging relevance in the non-simultaneous setting, assuming rational players obtain a greater payoff from misleading other players compared to learning the secret. Soundness has been achieved in prior work [10] focusing on non-simultaneous communication as follows: before reconstruction begins, all players are given protocol-induced side information alongside their list of shares. They must assume that when a player aborts communication, the previous round was the revelation round. Even if a deviant player has aborted early, using this side information, honest players can check that they have the correct value after reconstruction. If not, they terminate the reconstruction altogether.

However, achieving soundness this way compromises fairness, as a deviant player can use the side-information to check whether they can abort early and learn the secret before honest players. The authors of [30] were the first to propose a fair RSS scheme that can tolerate *arbitrary* side-information, by proposing the use of time-delay encryption (TDE) [6,32]. The basic idea of a TDE scheme is to encrypt a message such that it can only be decrypted after a specific amount of time has elapsed. The scheme in [30] employs a cryptographic memory-bound function[2] (CMBF) [1,16] as a way to achieve time-delay in the recovery of an encrypted sub-share of the secret. The fairness of their scheme is restored by setting the runtime of rounds of the secret sharing scheme to be less than the time it takes to decrypt the encrypted shares. Thus, there is no way for a deviant player to learn anything about the secret during a reconstruction round before they must decide whether to abort communication. In addition, a proof of the sender's work in computing their message is sent. The scheme proposed in [10] builds upon [30], by encrypting shares (shares are computed using Shamir's SS scheme) using the CMBF and further splits the encrypted shares into sub-shares, distributed to players. During processing, players independently evaluate the encrypted sub-shares to obtain the encrypted share, decrypt and then reconstruct the polynomial to obtain the secret. They use a *specific* form of side-information, called a checking share, which is an actual share of the secret that players can use to confirm they have reconstructed the correct secret,

[2] A CMBF is a family of deterministic algorithms such that an efficiently generated key can decrypt the encrypted input, with a lower-bound on the number of memory-access steps to do so.

thus achieving soundness.[3] We note that the memory-bound running times of employing the MBF in [16], a cryptographic version of which is used for time-delay in [10,30], endows a high cost on the players who have to verify the proof of work from messages received by other players. In addition, the players sending the message can potentially perform less work than what is stated in their accompanying proof [15,36]. These drawbacks suggest that a better time-delay mechanism should be explored to guarantee fairness, that reduces verification costs of the communicated messages and\or increases computational efficiency for honest players obtaining the secret shares after the delay.

1.1 Our Contributions

In this paper, we improve on [10] and propose a RSS scheme achieving fairness and soundness in the non-simultaneous communication setting from a CPU-bound function, as opposed to a CMBF, namely a *homomorphic time-lock puzzle*.

Informally, a time-lock puzzle (TLP) [35] embeds a secret into a puzzle such that it cannot be decrypted until a certain amount of time $\mathcal{T}$ has elapsed. Characteristics of a TLP include fast puzzle generation and security against parallel algorithms, assuming the sequentiality of the underlying mathematical problem [35]. A *homomorphic* time-lock puzzle (HTLP) scheme evaluates puzzles homomorphically using some operation, without the evaluator knowing the secret shares encapsulated within the corresponding puzzles. The resulting puzzle output contains the homomorphic evaluation of the input puzzles, enabling a more efficient way for decryptors to obtain the final output solution, as they can solve just one puzzle rather than solving all of the puzzles individually with standard TLPs, and then evaluating a final solution.

In our scheme, the dealer splits the secret into shares and creates an additional share which is broadcast to all players, i.e., the checking share. The rest of the shares are split into sub-shares, embedded into HTLPs and distributed to the corresponding players in such a way that the HTLP scheme can reconstruct the share from them. Intuitively, the checking share is used to verify the soundness of the secret that players reconstruct, and the delay provided by the HTLP scheme is used to guarantee fairness in the presence of a checking share for players communicating non-simultaneously. More specifically, the HTLP scheme embeds the sub-shares into puzzles that cannot be decrypted before a round of communication in the reconstruction phase has finished. Fairness is achieved by setting each round of communication to have an upper time bound of $\mathcal{T}$. Thus, a player wishing to deviate from their prescribed strategy and quit communication will not be able to derive the secret before the end of the round, in which case, the other players realise the deviant player has quit and output the result of the previous rounds reconstruction. We show that even if a player quits in a round and manages to learn the secret, the only case in which they can do so results

[3] Note that [30] works under the assumption that players prefer everyone to obtain the correct output over misleading others, therefore soundness is not an issue that needs to be addressed.

in the honest players also learning the secret. Therefore there is no advantage in a player deviating from their prescribed strategy.

From our generic construction, which we show satisfies soundness and fairness, we provide a concrete instantiation using the multiplicative variant of the HTLP scheme proposed in [31]. The result is a concrete, efficient scheme whose security relies on standard assumptions.

We argue that our improvement on prior work is threefold: we base the time delay of the construction on CPU-bound functions, as opposed to CMBFs; we provide an efficiency gain by using HTLPs instead of TLPs, and our solution has inherent flexibility.

Basing the time-delay primitive on CPU-bound functions as opposed to memory-bound functions captures a more realistic, inexpensive way to implement a SS scheme construction. Processors are faster than memory and scale better; even more so, fast memory is considerably more expensive. In practice, it is easier to raise the computational requirements of a player than it is memory accesses, up to a point, as adding more processors to a computer is more accessible than making memory accesses faster. A justification for using MBFs in [10,30] is that disparities in the computational power of players can cause unfairness when using standard TLPs for time-delay. However, with reasonable assumptions on the CPU-power of players, this disparity is not significant.

Furthermore, we use a HTLP for time-delay, which requires less computational work on behalf of the players decrypting puzzles compared to using standard TLPs. This efficiency improvement means that the consequence of disparities in CPU-power becomes less significant. To see this, evaluating several puzzles homomorphically, and then solving just *one* puzzle, requires fewer computational steps than solving individual puzzles and evaluating a function over the outputs, as in [10].

Finally, the instantiation of our generic scheme can use any correct SS scheme with a suitable HTLP, dependent on the application. The HTLPs that we use, from [31], are adaptable in the following ways: different operators (linear, multiplicative, and XOR) can be used, we can augment the setup with puzzles of different time hardness parameters $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$ or have a reusable setup, in which the scheme remains efficiently computable.

We refer the reader to the full version of this paper [24] for the formal security analysis of our generic scheme and concrete instantiation using multiplicative-homomorphic TLPs and a multiplicative SS scheme. For the remainder of this paper, any reference to appendices is in relation to the full version of our paper.

2 Definitions and Modelling

2.1 Secret Sharing

Informally, a (t, n) secret sharing scheme (SS) involves a dealer D, some secret s, and a set $P = \{P_1, P_2, \ldots, P_n\}$ of n players. The dealer distributes shares of a secret s chosen according to an efficiently samplable distribution of the set of

secrets, labelled $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$, with security parameter λ. The key idea behind threshold SS is that no subset $t' < t$ of players in P can learn the secret s, including an adversary controlling t' players. Conversely, every subset $t' \geq t$ of players in P is capable of reconstructing s.

A SS protocol is composed of two phases, share and reconstruction. During the share phase, the dealer samples a secret s from $\mathcal{S}_\lambda$ and generates n shares from the secret being distributed to each player in P. The dealer does this non-interactively, using a share algorithm to generate the set of shares to be distributed. The dealer digitally signs (typically using information-theoretically secure MACs) and encrypts the shares before distributing them to individual players over a broadcast channel [4].

The reconstruction phase itself is composed of two parts: communication and processing. The communication phase has players interact by sending their share over the broadcast channel to every other player in P (if a broadcast channel is not available to parties, then they have to send their share to each of the other players separately). Once players have communicated, they can move to the processing phase where they embark on reconstructing the secret s from the shares that they have received. This is under the assumption that a sufficient number of shares have been sent and received from other players, and that players followed the protocol (correctness). If an insufficient amount of shares have been received, the secret cannot be reconstructed, so players output $\perp$. Any player taking part in reconstruction proceeds to output their result.

Threshold secret sharing schemes have been explored extensively, and were introduced independently by Shamir [37] and Blakley [5]: Shamir's scheme is based on polynomial interpolation over a finite field of prime order, and Blakley's scheme is based on the uniqueness of hyperplane intersection. Extending the work of [37], [11,13,40] propose multiplicative homomorphic secret sharing schemes based on polynomial interpolation over finite groups with respect to multiplication, that need not be of prime order (See Appendix B [24]).

Next, we recall the formal definition of a threshold secret sharing scheme, with the implicit assumption that the dealer has digitally signed the shares before distributing:

Definition 1 ((t, n) Secret Sharing). *Given a dealer D, a secret $s \in \mathcal{S}_\lambda$ for security parameter λ, and a set of n authorised players $P = \{P_1, \ldots, P_n\}$, a (t, n) secret sharing scheme is a tuple of three PPT algorithms ($\mathsf{Setup}, \mathsf{Share}, \mathsf{Recon}$) defined as follows:*

- **Share Phase**: D takes as input the secret s and performs the following steps non-interactively:
  1. $pp \leftarrow \mathsf{Setup}(1^\lambda)$ a probabilistic algorithm that takes as input security parameter 1^λ and outputs public parameters pp, which are broadcast to all players in P.

[4] Privacy and authentication of the distribution of shares is a standard cryptographic assumption in secret sharing schemes [34].

2. $\{s_1, \ldots, s_n\} \leftarrow \mathsf{Share}(pp, s)$ a probabilistic algorithm that takes as input the secret $s \in \mathcal{S}_\lambda$ and outputs n shares s_i, one for each player in P.
3. Distribute s_i to player P_i for every $i \in [n]$ over a secret, authenticated channel.

- **Reconstruction Phase**: Any player in $P = \{P_1, \ldots, P_n\}$ is able to take part in this phase.
  1. Communication:
     (a) Each player P_i sends their share s_i over a secure broadcast channel to all other players in P.
     (b) P_i checks that they have received $(t-1)$ or more shares. If so, they proceed to processing.[5]
  2. Processing:
     Once P_i has a set of t' shares labelled S', they independently do the following:
     (a) $\{s, \perp\} \leftarrow \mathsf{Recon}(pp, S')$ a deterministic algorithm that takes as input the set S' of t' shares and outputs the secret s if $t' \geq t$ or outputs abort $\perp$ otherwise.

A (t, n) threshold SS scheme needs to satisfy the properties of correctness and secrecy, whose definitions are provided in [24], Appendix A.3. Informally, correctness means that an honest execution of the scheme results in the true secret being output, except with negligible probability; and secrecy ensures that reconstruction with fewer shares than the threshold (t) results in abort ($\perp$) being output, except with negligible probability.

2.2 Rational Secret Sharing

Using game-theory notions, players are considered to be rational if they have a preference in the outcome of the reconstruction phase. In a rational secret sharing (RSS) scheme, a players strategy is to maximise their payoff from the outcome of the game. The strategy σ_i taken by each player P_i must be determined by the dealer in order to achieve a fair outcome. Observe that depending on the scheme, the strategies of players in P may be the same or different.

In Definition 1, players only participate in the reconstruction phase. Therefore, we define a RSS scheme by providing a definition of the reconstruction phase only.

Definition 2 ((t, n) Rational Secret Reconstruction [10]). *A reconstruction phase $\Gamma_{t,n}$ is defined by $\Gamma_{t,n} = (\Gamma, \overrightarrow{\sigma})$ where Γ is the game to be played by players during the reconstruction phase and $\overrightarrow{\sigma} = (\sigma_1, \cdots, \sigma_n)$ denotes the strategy profile of the players in P prescribed by the dealer D during the share phase for that scheme.*

The outcome of the phase for all players is defined by the n-dimensional vector

[5] Whilst not explicit in the definition, there is an upper bound on how long players can communicate their shares for. Therefore, at the end of their communication, if a player P_i has not obtained a sufficient number of shares, then they output $\perp$ at the end of the reconstruction phase.

$$\overrightarrow{\omega}((\Gamma, \overrightarrow{\sigma})_{t,n}) = (\omega_1, \ldots, \omega_n)$$

where ω_i refers to the outcome of the phase for player P_i.

The outcome ω_i alludes to whether player P_i learns the entirety of s, nothing of s, is mislead into learning a fake secret s' or aborts the reconstruction phase altogether ($\perp$). It is important to note that the outcome of the phase depends on the strategy of the player.

One of the fundamental properties of secret sharing is *fairness* [39], which guarantees that no player has an advantage in the protocol over other players. The following defines fairness in the context of a RSS scheme. We use the following notation for a deviating strategy σ'_i for player P_i, to signify when a player behaves in a different way to how they are meant to. That is, they do not follow the protocol. In addition, P_{-i} represents all players in P excluding player P_i, and σ_{-i} signifies the honest strategies of this set of $(n-1)$ players, P_{-i}.

Definition 3 (Fairness [10]). *The reconstruction phase $\Gamma_{t,n}$ is* completely fair *if for every arbitrary alternative strategy σ'_i followed by player P_i for some $i \in [n]$, there exists a negligible function μ in the security parameter λ such that the following holds:*

$$\Pr[\omega_i(\Gamma, (\sigma'_i, \sigma_{-i})) = s] \leq Pr[\omega_{-i}(\Gamma, (\sigma'_i, \sigma_{-i})) = s] + \mu(\lambda).$$

That is, the probability of player P_i learning the secret when they deviate from their prescribed strategy in phase $\Gamma_{t,n}$ (but all other players follow their prescribed strategies) is only ever negligibly more than the probability of the other players learning the secret too. Consequently, such a player has no real advantage in deviating from their strategy.

How do we ensure that players (despite any preferences they may have) are motivated to follow a strategy in the non-simultaneous setting? This is typically done by assuming that the strategies of players are in a computationally strict Nash equilibrium (or some other variant of a Nash equilibrium) [14,23,33]. This concept makes certain that if every player $P_i \in P$ believes all other players in P are following their prescribed strategy in the phase, then they have nothing to gain in deviating from their own strategy and are penalised in some way by deviating. In our construction, we need to ensure players strategies are in a computationally strict Nash equilibrium when they additionally have access to side-information related to the secret. We discuss this further in Appendix D.2 of the full version of this paper [24].

Another fundamental property of RSS is soundness. Simply put, soundness of the reconstruction phase output means that the probability of players following the scheme outputting an incorrect secret when another player deviates from their own strategy is negligible.

Definition 4 (Soundness [10]). *Reconstruction phase $\Gamma_{t,n}$ is* sound *if for every arbitrary alternative strategy σ'_i followed by player P_i for $i \in [n]$, there exists a negligible function μ in the security parameter λ such that the following holds:*

$$\Pr[\omega_{-i}(\Gamma, (\sigma_i', \sigma_{-i})) \notin \{s, \perp\}] \leq \mu(\lambda)$$

In our construction, as we shall see, we achieve this property by using a checking share, similarly to [10]. A checking share is an actual share of the secret, kept separate from the other shares and publicly broadcast to players. In order to formalise our scheme, discussed in Section 3, we recall the definition of a homomorphic time-lock puzzle (HTLP) [31], on which our construction relies.

2.3 Homomorphic TLPs

Informally, a time-lock puzzle (TLP) scheme embeds a secret into a puzzle such that it cannot be decrypted until a certain amount of time $\mathcal{T}$ has elapsed. The seminal work of [35] outlined the characteristics of a TLP:

- Fast puzzle generation: namely, the time t required to generate a puzzle $\mathcal{Z}$ must be $t << \mathcal{T}$, for a given (time) hardness parameter $\mathcal{T}$.
- Security against parallel algorithms: that is, the encapsulated secret s is disguised within the puzzle $\mathcal{Z}$ for circuits of depth $< \mathcal{T}$, regardless of the size of the circuit.

However, when the decryptor is faced with a significant number of puzzles to solve, a standard TLP scheme requires the decryptor to solve each individual puzzle, which could be very inefficient. Driven by this limitation [31] introduced the notion of a homomorphic TLP (HTLP), a scheme that compactly evaluates puzzles homomorphically.

Homomorphic time-lock puzzles are augmented TLPs allowing anyone to evaluate a circuit C over sets of puzzles $(\mathcal{Z}_1, \ldots, \mathcal{Z}_n)$ homomorphically using operation Ψ [6], without the evaluator necessarily knowing the secret values $(s_1, \ldots, s_n)$ encapsulated within the corresponding puzzles. The resulting output (a puzzle $\mathcal{Z}$) contains the circuit output $C(s_1, \ldots, s_n)$, and the hardness parameter $\mathcal{T}$ does not depend on the size of the circuit C that was evaluated (this is called compactness).

Definition 5 (HTLP [31]). *Let $C = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a class of circuits and let secret space $\mathcal{S}_\lambda$ be a finite domain for security parameter λ. A homomorphic time-lock puzzle (HTLP) with respect to C and $\mathcal{S}_\lambda$ is defined by a tuple of four PPT algorithms ($\mathsf{HP.Setup}$, $\mathsf{HP.Gen}$, $\mathsf{HP.Solve}$, $\mathsf{HP.Eval}$) as follows:*

- $pp \leftarrow \mathsf{HP.Setup}(1^\lambda, \mathcal{T})$ is a probabilistic algorithm that takes as input security parameter 1^λ and hardness parameter $\mathcal{T}$ and outputs public parameters pp.
- $\mathcal{Z} \leftarrow \mathsf{HP.Gen}(pp, s)$ a probabilistic algorithm that takes as input the public parameters pp and a secret $s \in \mathcal{S}_\lambda$ and outputs a puzzle $\mathcal{Z}$.
- $s \leftarrow \mathsf{HP.Solve}(pp, \mathcal{Z})$ is a deterministic algorithm that takes as input public parameters pp and puzzle $\mathcal{Z}$, and outputs a solution s.

[6] What Ψ is depends on the application the HTLP is being used for. It could be addition, multiplication or XOR for example.

– $\tilde{\mathcal{Z}} \leftarrow \mathsf{HP.Eval}(pp, C, \Psi, \mathcal{Z}_1, \ldots, \mathcal{Z}_n)$ is a probabilistic algorithm taking as input a circuit $C \in \mathcal{C}_\lambda$, parameters pp, homomorphic-operation Ψ, and a set of n puzzles $(\mathcal{Z}_1, \ldots, \mathcal{Z}_n)$, and outputs a master puzzle $\tilde{\mathcal{Z}}$.

A HTLP scheme should satisfy correctness, security, and compactness. Informally, correctness means that if a scheme is executed properly, then the probability of the output being anything other than the solution is negligible. Captured within the definition of the correctness of a HTLP scheme [31] is the time-delay in solving a HTLP puzzle. Informally, given a puzzle evaluated in the scheme, there exists a fixed polynomial over the security and time hardness parameters which bounds the runtime solving the puzzle in the HTLP scheme.

Intuitively, a scheme is considered secure if the output of execution is indistinguishable from random to an eavesdropping adversary. Compactness is a non-trivial property requiring that the complexity of decrypting an evaluated ciphertext does not depend on the function used to evaluate the ciphertext. Intuitively, it means that the ciphertext size should not grow through homomorphic operations and the output length of the homomorphically evaluated ciphertext only depends on the security parameter. In the context of a HTLP, compactness therefore requires the size of the evaluated puzzle ciphertexts to be independent of the size of the circuit, and for the runtime of the evaluation algorithm to be independent of the hardness parameter $\mathcal{T}$.

3 A Fair and Sound Non-Simultaneous Rational Secret Sharing Scheme

We consider a RSS scheme with the reconstruction phase defined as in Definition 2. In our construction, the dealer runs the share phase, where they sample a value for the number of shares needed to reconstruct the secret, as well as splitting the secret into shares and further into sub-shares, similarly to the approach in [10]. Then, the dealer distributes a unique, ordered list of sub-shares to each player, alongside broadcasting public parameters.

The reconstruction phase works in rounds, with the n players in P performing the communication phase and processing phase in parallel. In the first round of the reconstruction phase, only the communication phase occurs. The processing phase does not start until the second round onwards. Each round (after the first) of the reconstruction phase works as follows. Players communicate (following the order of their given list) the sub-share corresponding to the round of Γ that they are in, one at a time. They must check at the end of the round that they have obtained sub-shares from all other players.

At the same time, players process the sub-shares received in the previous round, evaluating them over some function to obtain a share of the secret. After a certain number of rounds, as decided by the dealer, a sufficient number of shares will have been derived and players can use these shares to reconstruct the correct secret. The concept of rounds in RSS means that players gradually recover the secret, by reconstructing just one share per round, motivating all players to continue following the reconstruction phase.

More specifically, we let the dealer D be honest and non-interactive, only taking part in the share phase. Following [25], we assume that the dealer (information-theoretically) authenticates the shares distributed to players so that a player cannot send an incorrect share to another player, and the set of shares that a player sends to other players is unique. These assumptions translate to only one of two actions that a player can perform in each round: communicate (follow their strategy) or remain silent. We assume the players' strategies in the reconstruction phase are in a (computationally) strict Nash equilibrium in order to motivate them to follow the phase and not deviate.

In the share phase of our construction, D samples r, the revelation value. The revelation value signifies how many correctly run rounds, or equivalently, how many recovered shares are sufficient for a player to reconstruct the secret. D determines r by randomly sampling from an efficiently samplable discrete distribution $\mathcal{G}$, keeping the value secret from all players. Next, D obtains the first $(r+1)$ shares of the secret s; where the 0th share s_0 will be the checking share and is kept separate and broadcast to all players before the reconstruction phase. We note that the checking share is only used to verify the output of the reconstruction phase, and cannot be used to reconstruct the secret itself. This is necessary in order to ensure the soundness of the output.

Additionally, the dealer randomly samples a value d from an efficiently samplable discrete distribution $\mathcal{G}'$ and generates d fake shares, used to disguise the value r. Typically both $\mathcal{G}$ and $\mathcal{G}'$ are geometric distributions [18,10] (see Appendix C of [24]). Letting $m = r + d$, the dealer proceeds to create n sub-shares for each of the m shares, so that each player has a sub-share of every share, for a total of m sub-shares in each of the n lists, one for each player.

Similarly to [30], we need the sub-shares to be encrypted before being distributed to a player in a way that no player can decrypt their sub-shares before a round of communication is over. This is done so that players communicating non-simultaneously do not know until after they have broadcast their share for a given round, whether or not that was the revelation round. This is crucial to achieve fairness and ensure that players continue to be motivated to follow the scheme [21].

Our construction achieves this time delay using homomorphic time-lock puzzles (HTLPs), first introduced in [31] (see Definition 5). Using a HTLP scheme with hardness parameter $\mathcal{T}$, the dealer sets the time limit for each round of communication to be bounded above by time $\mathcal{T}$. Encrypting the sub-shares creates so-called sub-puzzles[7] of the sub-shares, which the dealer distributes as a list to individual players before reconstruction begins.

Each round of the reconstruction phase $\Gamma_{r,r+1}$ has players communicate non-simultaneously the corresponding sub-puzzle from their list, whilst processing in parallel the sub-puzzles received from the previous round. In a round of the

[7] We call the HTLP encryption of the sub-shares sub-puzzles for ease of understanding. They are simply time-lock puzzles that can be homomorphically evaluated to obtain a puzzle of the share which corresponds to the homomorphic evaluation of the given sub-shares.

communication phase, players must send their sub-share before time $\mathcal{T}$. Once this time has elapsed, a player checks that they have received $(n-1)$ sub-puzzles from the other players. In this case, in the next round of the reconstruction phase, these n sub-puzzles will be processed.

In the processing phase, players work independently and evaluate the n sub-puzzles from the previous round. In doing so, they will obtain a puzzle of the share for the previous round. This is computationally correct given that the sub-shares were derived by the dealer such that over some function the sub-shares homomorphically compute this share. The puzzle of the share is decrypted using the solve algorithm in the HTLP scheme to obtain the corresponding share.

Players attempt to reconstruct the actual secret from the shares that they have reconstructed so far. They determine whether they have reached the revelation round by using the checking share s_0 to confirm whether their solution is the real secret. If so, players output the secret s. If the reconstructed value as determined by the checking share, is $s' \neq s$, the players do not output a result. Instead, they will start the subsequent reconstruction phase round. Players repeat this cycle of steps until they have reconstructed the correct secret s unless either of the following scenarios occurs:

1. A deviant player has quit communicating in a round of the phase. Even if they correctly guess the right round to quit (round r), the time delay of the encrypted sub-puzzles ensures that the deviant player cannot decrypt the evaluated puzzle of the share before the end of a round.
   The non-deviant players quit communicating if at the end of the round they have received fewer than $(n-1)$ sub-puzzles. As a consequence, they cannot reconstruct a puzzle share for that round and will have an insufficient number of reconstructed shares, so the outcome for reconstruction will be $\perp$. The act of aborting means that no player learns the secret including the deviant player, as they are identified as a cheater before they can reconstruct the secret, if at all.
2. Players have sent the final, mth sub-puzzle from their list and so have no more sub-puzzles to share after this round. Players quit communication and attempt to reconstruct the secret from the shares that were reconstructed in the previous rounds.

3.1 Our Construction

Given an honest, non-interactive dealer D and a set of n rational players $P = \{P_1, \ldots, P_n\}$ communicating non-simultaneously, we use a HTLP to build a fair RSS scheme with sound output. Assume that each round of the reconstruction phase is bounded by the time hardness parameter $\mathcal{T}$.

Definition 6 (Non-Simultaneous RSS Scheme).

Given security parameter λ, time hardness parameter $\mathcal{T}$, an efficiently samplable distribution of the set of secrets $\mathcal{S}_\lambda$ with operator Ψ, secret $s \in \mathcal{S}_\lambda$, efficiently

samplable discrete distributions $\mathcal{G}, \mathcal{G}'$, we construct a RSS scheme with reconstruction phase in the non-simultaneous setting as a tuple of three PPT algorithms $(\mathsf{Setup}', \mathsf{Share}', \mathsf{Recon}')$ from a secret sharing scheme $(\mathsf{Setup}, \mathsf{Share}, \mathsf{Recon})$ and a HTLP scheme
$(\mathsf{HP.Setup}, \mathsf{HP.Gen}, \mathsf{HP.Solve}, \mathsf{HP.Eval})$ as follows:

- **Sharing Phase:** The honest dealer D takes as input the secret $s \in \mathcal{S}_\lambda$ and performs the following steps non-interactively:
  1. $pp' \leftarrow \mathsf{Setup}'(1^\lambda, \mathcal{T})$ a probabilistic algorithm on inputs $1^\lambda, \mathcal{T}$ in which the dealer runs:
     (a) $pp_1 \leftarrow \mathsf{HP.Setup}(1^\lambda, \mathcal{T})$ which outputs public parameters pp_1.
     (b) $pp_2 \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{T})$ which outputs the public parameters pp_2. Additionally let for $r \leftarrow_\$ \mathcal{G}$ be the sampled revelation value and d be a random value $d \leftarrow_\$ \mathcal{G}'$.
     Outputs are sampled values r, d and public parameters $pp' := \{pp_1, pp_2\}$.
  2. $\{s_0, \{list_1, \ldots, list_n\}\} \leftarrow \mathsf{Share}'(pp', s)$: a probabilistic algorithm that takes as input the secret $s \in \mathcal{S}_\lambda$ and public parameters pp'. The output consists of a checking share s_0 and lists labelled $list_j$ for $j \in [n]$, each composed of m sub-puzzles for $m = r + d$.
     (a) Run $\{s_0, \{s_1, \ldots, s_r\}\} \leftarrow \mathsf{Share}(pp_2, s)$ a probabilistic algorithm with inputs the public parameters pp_2 and secret $s \in \mathcal{S}_\lambda$. The outputs are $(r+1)$ shares of the secret; the checking share s_0 and s_i for $i \in [r]$.
     (b) $\{s_{r+1}, \ldots, s_m\} \leftarrow_\$ \mathcal{S}_\lambda$, randomly sample d fake shares from $\mathcal{S}_\lambda$.
     (c) For every $i \in [m]$, compute the list of sub-shares $\{s_{i,1}, \ldots, s_{i,n}\}$ such that $s_i = \underset{j \in [n]}{\Psi} s_{i,j}$.
     (d) Run $\mathcal{Z}_{i,j} \leftarrow \mathsf{HP.Gen}(pp_1, s_{i,j})$ a probabilistic algorithm that takes as input sub-shares $s_{i,j}$ and public parameters pp_1, and outputs sub-puzzles $\mathcal{Z}_{i,j}$, $\forall i \in [m], \forall j \in [n]$.
     (e) D distributes $\text{list}_j = \{\mathcal{Z}_{1,j}, \cdots, \mathcal{Z}_{r,j}, \mathcal{Z}_{r+1,j}, \cdots, \mathcal{Z}_{m,j}\}$ to the corresponding player P_j, for every $j \in [n]$.
  3. The dealer distributes the following:
     (a) D broadcasts $\{pp', s_0\}$ to all P the public parameters pp' and checking share s_0.
     (b) D distributes list_j to P_j for every $j \in [n]$.

- **Reconstruction Phase:** All players in $P = \{P_1, \ldots, P_n\}$ independently take part in this phase.

  1. Communication: We are in the kth round of the communication, for some $1 < k \leq m$.
     (a) P_j sends to all of P the sub-puzzle $\mathcal{Z}_{k,j}$ for every $j \in [n]$ non-simultaneously.
     (b) At the end of round k (after time $\mathcal{T}$ has elapsed), along with their own sub-puzzle, player P_j should have received $\{\mathcal{Z}_{k,1}, \ldots, \mathcal{Z}_{k,n}\}$ from all of P.

(c) Move to round $(k+1)$ of communication and round k of processing, unless fewer than $(n-1)$ sub-puzzles have been received. In this case, proceed to abort communication and move to 2c with reconstructed shares $\{s_1, \ldots, s_{k-1}\}$.

2. Processing: We are in round $(k-1)$ of processing, for some $1 < k \leq m$.[8] For any $j \in [n]$, P_j does the following:
   (a) $\mathcal{Z}_{k-1} \leftarrow \mathsf{HP.Eval}(pp_1, \mathcal{T}, \Psi, \mathcal{Z}_{k-1,1}, \cdots, \mathcal{Z}_{k-1,n})$: Run the probabilistic algorithm $\mathsf{HP.Eval}$ with inputs the public parameters pp_1, hardness parameter $\mathcal{T}$, and the list of n sub-puzzles for the $(k-1)$th round, a player homomorphically evaluates sub-puzzles with operator Ψ to output share puzzle $\mathcal{Z}_{k-1}$.
   (b) $s_{k-1} \leftarrow \mathsf{HP.Solve}(pp_1, \mathcal{T}, \mathcal{Z}_{k-1})$: Run the probabilistic algorithm $\mathsf{HP.Solve}$ that takes as input the public parameters pp_1; hardness parameter $\mathcal{T}$; and puzzle share $\mathcal{Z}_{k-1}$ and outputs secret share s_{k-1}. Output the round share s_{k-1} and move to reconstructing s.
   (c) $\{s, \perp\} \leftarrow \mathsf{Recon}'(pp', s_0, \{s_1, \ldots, s_{k-1}\})$: where the players run $\{s, \perp\} \leftarrow \mathsf{Recon}(pp_2, \{s_1, \ldots, s_m\})$, a deterministic algorithm that inputs public parameters pp_2 and $(k-1)$ reconstructed shares of the secret $\{s_1, \ldots, s_{k-1}\}$. Player P_j uses checking share s_0 to confirm the soundness of their reconstructed value and outputs either the correct secret s or abort $\perp$.
   (d) If P_j outputs $\perp$, but no player quit in round k of communication and every player $P_j \in P$ has $list_j \neq \emptyset$, then players go to $(k+1)$th round of reconstruction phase. If either case holds, output $\perp$.

We have the following result.

Theorem 1. *Our non-simultaneous rational secret scheme* $(\mathsf{Setup}', \mathsf{Share}', \mathsf{Recon}')$ *satisfies correctness, fairness and soundness in the presence of side information related to the secret, assuming the following properties:*

- *correctness, security, and compactness of the HTLP scheme,*
- *correctness and secrecy of the SS scheme,*
- *the checking share side information is correct, protocol-induced auxiliary information.*

We prove Theorem 1 in Appendix F of [24], demonstrating that our construction satisfies correctness, achieves soundness in the non-simultaneous setting using protocol-induced side information, and achieves fairness despite the presence of this side-information by using a HTLP to provide a time-delay to the scheme.

More specifically, in our security analysis, we summarise the scenarios in which a deviant player attempts to mislead. In particular, we demonstrate that if a player aborts in a round k with respect to revelation round r, regardless of the round that k is, the outcome for all players is the same. Analysing the scenarios in which a players quits communicating aids the proofs of fairness

[8] At least one round of communication is required before players can start processing.

and correctness, by providing an intuition to the outcome of the reconstruction phase.

Fairness of the scheme is proven as follows: we show that Definition 3 is satisfied in our construction assuming the correctness and security of the HTLP scheme [31] (definitions of which are provided in [24], Appendix A.2), which is employed to implement a time-delay in the scheme. We use a reduction to break the correctness and security of the HTLP scheme, contradicting our assumptions, in order to show that there does not exist a deviant player with the ability to decrypt a puzzle in time less than $\mathcal{T}$. Furthermore, assuming the correctness and secrecy of the underlying SS scheme (see Appendix A.3 [24]), we show that the probability of a deviant player learning the secret, whilst other players do not, is negligible in the security parameter λ. Observe that we additionally show in [24] that the rational players strategies $\overrightarrow{\sigma}$ are in a computationally strict Nash equilibrium (Appendix D.2, Definition 16 [24]) following the proofs of [10,30].

In order to prove soundness, we provide an Appendix E preceding the analysis of Theorem 1 in the full version of our paper [24], to define the side information used to achieve soundness. We closely follow the proof of [10] by firstly defining a membership oracle. Informally, this is an oracle queried by players in reconstruction in order to check the soundness of their reconstructed value [30]. Following [10], we claim and prove that the checking share in our construction can be used in place of a sound membership oracle (see Definition 18, Appendix E [24]), as a specific form of protocol-induced side information to ensure soundness. Finally, we prove that our generic construction achieves soundness with a checking share (found in Appendix F, Theorem 2 of[24]).

We defer the reader to [24], Appendix F for the full details of our proofs.

Next, we highlight the efficiency improvements our construction achieves by using a HTLP over standard TLPs. We then discuss how our results improve upon the scheme of [10], the most relevant related work.

HTLPs vs. TLPs The homomorphic property of a HTLP scheme means that solving a puzzle, the most computationally expensive step for the players, need only be run once rather than n times in the processing phase of our scheme. The computational cost of running $\mathsf{HP.Solve}$ is $\Omega(2^{\mathcal{T}})$-steps [9].

Indeed, if we were to use a standard TLP in the processing phase of our scheme, each player would independently have to solve each of the n sub-puzzles using $\mathsf{P.Solve}$, and then evaluate the n sub-shares to obtain the share for that round. Conversely, by using a HTLP in our scheme, players must run $\mathsf{HP.Eval}$ *once* over the n sub-puzzles, outputting a master puzzle, and proceed to run $\mathsf{HP.Solve}$ *once* on this master puzzle to obtain the corresponding share. Thus, HTLPs are more efficient by a linear factor of n, where n corresponds to the number of players participating in the reconstruction phase.

It is important that the homomorphic property of the HTLP scheme satisfies the definition of compactness in [31] (found in the full version of this paper [24],

[9] In a standard TLP scheme, the computational complexity of the puzzle-solving algorithm $\mathsf{P.Solve}$ is the same as $\mathsf{HP.Solve}$.

Appendix A.2, Definition 9). This means that the runtime of homomorphically evaluating puzzles, is bounded above by a fixed polynomial that only depends on the security parameter λ and not the time hardness parameter $\mathcal{T}$. Otherwise, the trivial solution would be indeed to use a standard TLP scheme.

Comparison with [10] Our scheme closely follows the work of [10]. Their construction involves linearly evaluating sub-shares encrypted using memory-bound functions for the time-delay to ensure fairness of the scheme and reconstructing the secret using Shamir's SS scheme. In contrast, our generic construction uses CPU-bound HTLPs to ensure a time-delay in rounds of the scheme, which we have argued in the Introduction constitutes an improvement.

Furthermore, the construction of [10] requires players to independently decrypt each share before they proceed to the secret reconstruction using Shamir SS scheme. The advantage of using HTLPs is that they provide an efficiency improvement for the honest players evaluating puzzles in comparison to using standard TLPs. Therefore our contributions are the efficiency improvements for honest players in homomorphically evaluating puzzles.

Finally, we have generalised our construction so that it can be adapted for different applications. The HTLP schemes of [31] are flexible in using different homomorphic operations and can be extended to using puzzles with varying levels of hardness (different $\mathcal{T}$ values), with potential for public-coin setup schemes and reusable setup schemes. Unlike [10] who provide a concrete scheme, our construction is generic and adaptable to the application for which it is being used.

3.2 A Concrete Instantiation

Our final contribution is to provide a concrete fair and sound RSS scheme by instantiating our construction with a specific variant of Shamir's SS scheme and a multiplicative HTLP (MHTLP [31]). In more detail, we instantiate our construction as follows:

- A multiplicative homomorphic threshold secret sharing scheme $(\mathsf{Setup}, \mathsf{Share}, \mathsf{Recon})$ (Appendix B of [24]), for a secret space $\mathcal{S}_\lambda$ over a finite group with respect to multiplication, defined as in [40,13,11],
- A MHTLP scheme $(\mathsf{MHP.Setup}, \mathsf{MHP.Gen}, \mathsf{MHP.Eval}, \mathsf{MHP.Solve})$ (in [24], Appendix B), which is multiplicatively homomorphic over a ring $(\mathbb{J}_N, \cdot)$.

The multiplicative operator $\otimes$ enables the dealer to split the ith share, for some $i \in [m]$, of the secret into n sub-shares in the following way,

$$s_{i,n} = s_i \cdot \left(\prod_{j=1}^{n-1} s_{i,j} \right)^{-1},$$

enabling players to homomorphically evaluate sub-puzzles by running $\mathsf{MHP.Eval}$, and $\mathsf{MHP.Solve}$ on the master puzzle output from evaluation to obtain the correctly reconstructed share for the ith round. To ensure soundness of the concrete instantiation, the dealer distributes a checking share s_0 to all players.

This is computed as $s_0 = f(y_0) \pmod N$, for some polynomial f determined in the setup phase of the scheme from a multiplicative homomorphic threshold SS scheme.

Whilst we have not implemented the concrete instantiation here, we note the following: firstly, the inclusion of a MHTLP scheme to a secret sharing scheme does increase the computational burden on the players participating in reconstruction, however it provides the important property of fairness in our scheme when soundness is additionally being provided by the means of side information. Secondly, we use a multiplicative-homomorphic TLP rather than a standard TLP in order to reduce the computational overhead for players by a linear factor. Indeed, in the instantiation, one run of $\mathsf{MHP.Eval}$ is necessary, which translates to n multiplications. This is followed by one run of $\mathsf{MHP.Solve}$ of complexity $\Omega(2^{\mathcal{T}})$. If we used a plain TLP in the instantiation instead, assuming the same parameters, we require n runs of $\mathsf{HP.Solve}$ of complexity $\Omega(2^{\mathcal{T}})$, followed by n runs of $\mathsf{HP.Eval}$, which means n multiplications.

In addition to the assumptions used in the security analysis of our generic construction, the instantiation relies on standard cryptographic and number-theoretical assumptions, including the sequential squaring and decisional Diffie-Hellman assumptions for a MHTLP [31], found in Appendix A.2 of [24]. We defer to [24] (Appendix C) for a full description of the instantiation of our construction.

Final remarks In this paper we have proposed a construction for a fair and sound rational secret sharing scheme in the non-simultaneous setting of communication from homomorphic time-lock puzzles. We have argued the benefits of this novel approach, and we have suggested a concrete scheme, relying on standard assumptions.

References

1. M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately hard, memory-bound functions. *ACM Transactions on Internet Technology*, 5:299–327, 2005.
2. G. Asharov and Y. Lindell. Utility dependence in correct and fair rational secret sharing. In S. Halevi, editor, *Lecture Notes in Computer Science*, volume 5677, pages 559–576. Annual International Cryptology Conference, CRYPTO 2009, Springer, 2009.
3. A. Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Technion-Israel Institute of technology, Faculty of computer science, 1996.
4. A. Beimel. Secret-sharing schemes: A survey. In Y.M. Chee et al., editors, *Lecture Notes in Computer Science*, volume 6639, pages 11–46. International Conference on Coding and Cryptology, Springer, 2011.
5. G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the AFIPS National Computer Conference, NCC 1979*, volume 48, pages 313–318. International Workshop on Managing Requirements Knowledge (MARK), IEEE, 1979.
6. J. Cathalo, B. Libert, and J. Quisquater. Efficient and non-interactive timed-release encryption. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Lecture Notes in Computer Science*, volume 3783, pages 291–303. International Conference on Information and Communications Security, ICICS 2005, Springer, 2005.

7. D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In C. Pomerance, editor, *Lecture Notes in Computer Science*, volume 293, pages 11–19. Advances in Cryptology- CRYPTO 1987, Springer, 1988.
8. R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, page 364–369. STOC 1986, Association for Computing Machinery, 1986.
9. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In B. Preneel, editor, *Lecture Notes in Computer Science*, volume 1807, pages 316–334. Advances in Cryptology- EUROCRYPT 2000, Springer, 2000.
10. S. J. De and Asim K. Pal. Achieving correctness in fair rational secret sharing. In M. Abdalla, Cristina N. R., and R. Dahab, editors, *Lecture Notes in Computer Science*, volume 8257, pages 139–161. International Conference on Cryptology and Network Security, CANS 2013, Springer, 2013.
11. Y. Desmedt, G. Di Crescenzo, and M. Burmester. Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In J. Pieprzyk and R. Safavi-Naini, editors, *Lecture Notes in Computer Science*, volume 917, pages 19–32. Advances in Cryptology, ASIACRYPT 1994, Springer, 1994.
12. Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Lecture Notes in Computer Science*, volume 576, pages 457–469. Advances in Cryptology- CRYPTO 1991, Springer, 1991.
13. Y. G. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM journal on Discrete Mathematics*, 7(4):667–679, 1994.
14. Y. Dodis and T. Rabin. Cryptography and game theory. *Algorithmic Game Theory*, pages 181–207, 2007.
15. S. Doshi, F. Monrose, and A. D. Rubin. Efficient memory bound puzzles using pattern databases. In J. Zhou, M. Yung, and F. Bao, editors, *Lecture Notes in Computer Science*, volume 3989, pages 98–113. International Conference on Applied Cryptography and Network Security, ANCS 2006, Springer, 2006.
16. C. Dwork, A. Goldberg, and M. Naor. On memory-bound functions for fighting spam. In D. Boneh, editor, *Lecture Notes in Computer Science*, volume 2729, pages 426–444. Advances in Cryptology, CRYPTO 2003, Springer, 2003.
17. G. Fuchsbauer, J. Katz, and D. Naccache. Efficient rational secret sharing in standard communication networks. In D. Micciancio, editor, *Lecture Notes in Computer Science*, volume 5978, pages 419–436. Theory of Cryptography Conference, TCC 2010, Springer, 2010.
18. S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. *Journal of the ACM (JACM)*, 58(6):1–37, 2011.
19. S. D. Gordon and J. Katz. Rational secret sharing, revisited. In *Lecture Notes in Computer Science*, volume 4116, pages 229–241. International Conference on Security and Cryptography for Networks, SCN 2006, Springer, 2006.
20. V. Goyal, O. Pandey, and B. Sahai, A.and Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, page 89–98. CCS 2006, Association for Computing Machinery, 2006.
21. 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 623–632. STOC 2004, Association for Computing Machinery, 2004.

22. L. Harn, C. Lin, and Y. Li. Fair secret reconstruction in (t, n) secret sharing. *Journal of Information Security and Applications*, 23:1–7, 2015.
23. J. Katz. Bridging game theory and cryptography: Recent results and future directions. In R. Canetti, editor, *Lecture Notes in Computer Science*, volume 4948, pages 251–272. Theory of Cryptography Conference, TCC 2008, Springer, 2008.
24. J. Knapp and E.A. Quaglia. Fair and sound secret sharing from homomorphic time-lock puzzles. Cryptology ePrint Archive, Report 2020/1078, 2020. `https://eprint.iacr.org/2020/1078`.
25. G. Kol and M. Naor. Cryptography and game theory: Designing protocols for exchanging information. In R. Canetti, editor, *Lecture Notes in Computer Science*, volume 4948, pages 320–339. Theory of Cryptography Conference, TCC 2008, Springer, 2008.
26. G. Kol and M. Naor. Games for exchanging information. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC 2008*, page 423–432. Association for Computing Machinery, 2008.
27. H. Krawczyk. Secret sharing made short. In *Lecture Notes in Computer Science*, volume 773, pages 136–146. Advances in Cryptology, CRYPTO 1993, Springer, 1993.
28. C-S Laih and Y-C Lee. V-fairness (t, n) secret sharing scheme. *IEE Proceedings-Computers and Digital Techniques*, 144(4):245–248, 1997.
29. H-Y Lin and L. Harn. Fair reconstruction of a secret. *Information Processing Letters*, 55(1):45–47, 1995.
30. A. Lysyanskaya and A. Segal. Rational secret sharing with side information in point-to-point networks via time-delayed encryption. *IACR Cryptology ePrint Archive*, 2010:540, 2010.
31. G. Malavolta and S. A. K. Thyagarajan. Homomorphic time-lock puzzles and applications. In A. Boldyreva and D. Micciancio, editors, *Lecture Notes in Computer Science*, volume 11692, pages 620–649. Annual International Cryptology Conference, CRYPTO 2019, Springer, 2019.
32. T. C. May. Time-release crypto. In *Manuscript*, 1993.
33. J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.
34. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Lecture Notes in Computer Science*, volume 576, pages 129–140. Advances in Cryptology, CRYPTO 1991, Springer, 1991.
35. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. *Technical Report MIT/LCS/TR-684*, 1996.
36. D. Rosenthal. On the cost distribution of a memory bound function. *arXiv preprint cs/0311005*, 2003.
37. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
38. Y. Tian, J. Ma, C. Peng, and J. Zhu. Secret sharing scheme with fairness. In *10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 494–500. IEEE, 2011.
39. M. Tompa and H. Woll. How to share a secret with cheaters. *Journal of Cryptology*, 1(3):133–138, 1989.
40. H. Wang, K.Y. Lam, G.Z Xiao, and H. Zhao. On multiplicative secret sharing schemes. In E.P. Dawson, A. Clark, and C. Boyd, editors, *Lecture Notes on Computer Science*, volume 1841, pages 342–351. Information Security and Privacy, ACISP 2000, Springer, 2000.

41. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In D. Catalano, N. Fazio, R. Gennaro, and Nicolosi A., editors, *Lecture Notes in Computer Science*, volume 6571, pages 53–70. International Workshop on Public Key Cryptography- PKC 2011, Springer, 2011.