An Efficient Rational Secret Sharing Scheme Based on the Chinese Remainder Theorem

Yun Zhang[1,2], Christophe Tartary[3], and Huaxiong Wang[1]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
ZHAN0233@e.ntu.edu.sg

[2] School of Mathematical Science, Yangzhou University, Yangzhou, 225002, People's Republic of China
ctartary@mail.tsinghua.edu.cn

[3] Institute for Interdisciplinary Information Sciences, Institute for Theoretical Computer Science, Tsinghua University, Beijing, 100084, People's Republic of China
hxwang@ntu.edu.sg

Abstract. The design of rational cryptographic protocols is a recently created research area at the intersection of cryptography and game theory. At TCC'10, Fuchsbauer *et al.* introduced two equilibrium notions (computational version of strict Nash equilibrium and stability with respect to trembles) offering a computational relaxation of traditional game theory equilibria. Using trapdoor permutations, they constructed a rational t-out-of n sharing technique satisfying these new security models. Their construction only requires standard communication networks but the share bitsize is $2n|s|+O(k)$ for security against a single deviation and raises to $(n-t+1)\cdot(2n|s|+O(k))$ to achieve $(t-1)$-resilience where k is a security parameter. In this paper, we propose a new protocol for rational t-out-of n secret sharing scheme based on the Chinese reminder theorem. Under some computational assumptions related to the discrete logarithm problem and RSA, this construction leads to a $(t-1)$-resilient computational strict Nash equilibrium that is stable with respect to trembles with share bitsize $O(k)$. Our protocol does not rely on simultaneous channel. Instead, it only requires synchronous broadcast channel and synchronous pairwise private channels.

Keywords: rational cryptography, computational strict Nash equilibrium, stability with respect to trembles, Asmuth-Bloom sharing scheme.

1 Introduction

1.1 Preliminaries

In 1979, Shamir [16] and Blakley [4] independently introduced the concept of *secret sharing scheme* (SSS) in order to facilitate the distributed storage of private data in an unreliable environment. Since then, secret sharing has become a major building block for cryptographic primitives in particular in the area of *multiparty computation* (MPC). The goal of a (perfect) SSS is to distribute a secret value s amongst a finite set of participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ in such a way that only specific subsets of $\mathcal{P}$ can reconstruct s while the others have no information about this secret element whatsoever.

Traditional cryptographic models assume that some parties are honest (i.e. they faithfully follow a given protocol) while others are malicious participants against whom the honest players must be protected. However, in many real-world applications, a participant will choose to be dishonest if deviating from the protocol will provide him with some advantage. Game theory can be used to model such a situation where players are *self-interested* (i.e. *rational*). In this representation, each participant P_i has a utility function U_i and the execution of the cryptographic protocol is regarded as a game over $\mathcal{P}$ where the n players' strategies $\sigma_1, \ldots, \sigma_n$ are dictated by their respective utilities $U_1, \ldots, U_n$.

Halpern and Teague introduced the first general approach for rational secret sharing in 2004 [9]. This opened new research directions and many results appeared subsequently [6,1,7,12,13,10,14,2]. In game theory, a *Nash equilibrium* (NE) captures the idea of stable solution for a given game. Indeed, in a NE, no single player P_i can individually improve his welfare by deviating from the strategy σ_i specified by the equilibrium $(\sigma_1, \ldots, \sigma_n)$ if all remaining participants stick to theirs. Most of the rational protocols quoted above focus on achieving a NE surviving iterated deletion of weakly dominated strategies. However, as pointed out in [12], some bad strategies still survive this deletion process. As a remedy, Kol and Naor proposed to use the notion of *strict NE* requiring that each player's strategy is his unique best response to the other players' strategies. This notion is more appealing than a NE in that, in a NE, there is no incentive to deviate while, in a strict NE, there is an incentive *not* to deviate. However, it is difficult to achieve a strict NE in many cases since this notion rules out many cryptographic techniques. In order to balance this tradeoff, Fuchsbauer *et al.* [7] proposed a *computational version of strict NE* (which enables the use of cryptography) and the notion of NE *stable with respect to trembles.* They also provided an efficient construction for standard communication networks achieving such an equilibrium as long as all the players are computationally bounded. However, the bitlength of their shares is $2n|s| + O(k)$ which gets very large especially when n (number of players) or k (security parameter) is large. While not a serious issue in its own right, this may be problematic when their rational SSS is used as a subroutine for rational MPC.

1.2 Our Results

In this paper, we present a protocol for rational t-out-of-n SSS. We only need a synchronous (but non-simultaneous) broadcast channel along with pairwise point-to-point channels. We do not assume any on-line dealer nor do we apply any generic MPC protocol to redistribute the shares of the secret. Instead, we borrow the idea from *Joint Random Secret Sharing* to allow every player to form his "one-time" share at the beginning of each iteration by interactions among the group of $m (m \geq t)$ participants. The main idea is described as follows.

In the share distribution phase, the dealer use the modified version of Asmuth-Bloom SSS proposed by Kaya and Selçuk [11] to generate n shares for the secret s. Suppose there are m players active in the reconstruction phase, say $P_1, \ldots, P_m$. This phase proceeds with several rounds. At the beginning of each iteration, the "one-time" shares for $(s+d) \bmod m_0$ are generated (jointly by the active players) using the technique from *Joint Random Secret Sharing*, where $d = d^{(1)} + \cdots + d^{(m)}$ and each $d^{(i)}$ is chosen independently and uniformly at random from the domain of the secret by P_i. If $d \equiv 0 \bmod m_0$, then all the "one-time" shares are valid for recovering s and, in this sense, the current iteration is called the valid iteration. Otherwise, the current iteration is invalid, which is designed only for catching possible cheaters. Each communicated message carries a commitment with perfect binding and computational hiding (assuming the hardness of computing discrete logarithm). Thus, at every point of our protocol, there is a unique legal message that each player can send (except with negligible probability). This prevents a player from outwardly appearing to follow the protocol while subliminally communicating with other participants.

Then, all the active players are required to open their "one-time" shares. After each player P_i has received the "one-time" shares from all the other active players, he is required to open $d^{(i)}$, which provides a unique way for the participants to jointly identify the valid iteration. If $d \bmod m_0 \neq 0$, then the current iteration is invalid and all the players are asked to restart a new iteration; otherwise, it is valid, the secret s is recovered and the protocol terminates immediately after this iteration. In this way, no player can identify the valid iteration before he opens his "one-time" share. Furthermore, each player can identify the valid iteration only after it has occurred, that is, once a player learns that the current iteration is valid, each player has already got the real secret. Due to this, we do not need simultaneous channels. Our protocol is efficient in that the round complexity and computation complexity are both polynomial (in the security parameter k). It induces a $(t-1)$-resilient computational strict Nash equilibrium that is stable with respect to trembles. However, our protocol relies on the assumption that no player knows auxiliary information about the secret s, which has been proved to be inherent in the non-simultaneous channels model [2].

1.3 Comparison to Fuchsbauer *et al.*'s Scheme

The protocol from [7] provides good point of comparison to ours since both techniques have similar features:

- Both of them induce a $(t-1)$-resilient computational strict NE that is stable with respect to trembles.
- Neither of them relies on simultaneous channels.
- Both of them assume that no player knows any auxiliary information about the secret s. This property has been proved to be inherent to the non-simultaneous channels model [2].
- Both protocols run in time polynomial in k (security parameter) and they have almost the same round complexity.

However, our protocol has smaller share size even when $(t-1)$-resilience to coalitions is required. Our shares are $O(k)$ bits long while those from [7] need $(n-t+1)(2n|s|+O(k))$ bits. The latter share length leads to practical efficiency issues when $n-t+1$ is large or when Fuchsbauer *et al.*'s technique is used as a building block within more general rational MPC protocols.

2 Definitions and Background

2.1 Secret Sharing

A t-out-of-n SSS with secret domain S is a two-phase protocol (share distribution and secret reconstruction) executed by the dealer and a subgroup of the n players $P_1, \ldots, P_n$ respectively. During the share distribution phase, the dealer chooses a secret $s \in S$ and generates n shares $s_1, \ldots, s_n$ based on a security parameter k. Each s_i is given to P_i secretly. In the secret reconstruction phase, some collection of at least t players jointly reconstruct s from their shares without any interaction with the dealer. We require the following two properties to hold:

- **Correctness.** Any collection of t or more players can uniquely determine the secret by putting their shares together honestly.
- **Privacy.** Any collection of fewer than t players can not recover the secret s.

In this paper, the security will be guaranteed under some computational assumptions related to the discrete logarithm problem and RSA which will be specified in Sect. 3.3. Thus, the security of our rational SSS will be computational.

2.2 Notions of Game-Theoretic Equilibria

As said in Sect. 1.1, in the rational model, each player is self-interested: he does what is in his interest. To formalize rationality, each player P_i is associated to a real-valued utility function U_i modeling the gain that P_i obtains when following his many strategies. For more details, we refer the reader to [1].

We now present the game theoretic concepts our cryptographic construction relies on. We are to design a rational SSS with the expectation that, when rationally played, the secret is revealed to all the players participating in the reconstruction. In the share distribution phase, all n players are silent and the dealer is assumed to be honest. The reconstruction process is to be viewed as a game amongst $m \geq t$ players. We denote $\sigma = (\sigma_1, \ldots, \sigma_m)$ the strategy profile of these players where σ_i is P_i's strategy for $1 \leq i \leq m$. As usual, let σ_{-C} denote the strategy profile of all m players except the players in C and σ_C denote the strategy profile constricted to the coalition $C \subseteq \{1, \ldots, m\}$. Given a strategy profile σ, it induces the utility value $U_i(\sigma)$ for each player P_i expressing his payoff when σ is played by the m players.

In the following, we denote the security parameter by k and it is assumed that the n utility functions are polynomials in k. The definitions appearing in this subsection originate from [7].

Definition 1. *Let $\epsilon : \mathbb{N} \rightarrow [0, \infty)$ be a function. We say ϵ is **negligible** if for every positive polynomial $p(\cdot)$ there exists an integer $N_{p(\cdot)} > 0$ such that for all $k > N_{p(\cdot)}$, it holds that $\epsilon(k) < \frac{1}{p(k)}$. We say that ϵ is **noticeable** if there exists a positive polynomial $p(\cdot)$ and an integer $M_{p(\cdot)}$ such that $\epsilon(k) > \frac{1}{p(k)}$ for any $k > M_{p(\cdot)}$.*

Definition 2. *A strategy σ induces an r-**resilient computational NE** if for any coalition C of at most r players and for any probabilistic polynomial time strategy profile σ', it holds:*

$$U_i(k, \sigma'_C, \sigma_{-C}) \leq U_i(k, \sigma_C, \sigma_{-C}) + \epsilon(k) \quad \textit{for any } i \in C,$$

where ϵ is a negligible function.

Remark 1. When $r = 1$, the definition of r-resilient computational NE coincides with that of the computational NE.

We need to define what it means for two strategies to be equivalent. Although we could refer the reader to [7] for the details, for completeness of our paper, we recall the corresponding notions below. As said before, every player is to be considered as a polynomial-time probabilistic Turning (PPT) machine (as function of the security parameter k). We assume that m players participate in the reconstruction phase. As often in MPC, security will be demonstrated by simulating the views of the different participants [8].

Definition 3. *Denote $P_C := \{P_i | i \in C\}$, $P_{-C} := \{P_i | i \notin C\}$ and the strategy vector of the m players by σ. Define the random variable $\mathsf{View}^{\sigma}_{-C}$ as follows:*

> *Let* Trans *denote the messages sent by P_C* not including any message sent by P_C after they write to their output tapes. $\mathsf{View}^{\sigma}_{-C}$ *includes the information given by the dealer to P_{-C}, the random coins of P_{-C} and the (partial) transcript* Trans.

Fix a strategy ρ_C and an algorithm T. Define the random variable $\mathsf{View}^{T,\rho_C}_{-C}$ as follows:

> *When the m players interact, P_C follows ρ_C and P_{-C} follows σ_{-C}. Let* Trans *denote the messages sent by P_C. Algorithm T, given the entire view of P_C, outputs an arbitrary truncation* Trans' *of* Trans *(defining a cut-off point and deleting any messages sent after that point).* $\mathsf{View}^{T,\rho_C}_{-C}$ *includes the information given by the dealer to P_{-C}, the random coins of P_{-C}, and the (partial) transcript* Trans'.

Strategy ρ_C yields equivalent play with respect to σ, *denoted $\rho_C \approx \sigma$, if there exists a PPT algorithm T such that for all PPT distinguishers D:*

$$\left| \mathrm{Prob}[D(1^k, \mathsf{View}^{T,\rho_C}_{-C}) = 1] - \mathrm{Prob}[D(1^k, \mathsf{View}^{\sigma}_{-C}) = 1] \right| \leq \epsilon(k)$$

where $\epsilon(\cdot)$ is a negligible function.

Definition 4. *A strategy σ is said to be an r-**resilient computational strict NE**, if:*

1. *σ induces an r-resilient computational NE;*
2. *For any coalition C of at most r players and for any probabilistic polynomial time strategy σ'_C with $\sigma'_C \not\approx \sigma$, there is a positive polynomial $p(\cdot)$ such that for any $i \in C$, it holds that $U_i(k, \sigma_C, \sigma_{-C}) \geq U_i(k, \sigma'_C, \sigma_{-C}) + \frac{1}{p(k)}$ for infinitely many values of k, namely, $U_i(k, \sigma_C, \sigma_{-C}) - U_i(k, \sigma'_C, \sigma_{-C})$ is non-negligible.*

Definition 5. *For any coalition C, strategy ρ_C is δ-**close** to strategy σ_C if ρ_C is as follows:*

ρ_C: With probability $1 - \delta$, players in C play according to σ_C.
*With probability δ, players in C follow an arbitrary (possibly correlated) PPT strategy σ'_C (called the **residual strategy** of ρ_C).*

Definition 6. *σ induces an r-**resilient computational NE that is stable with respect to trembles** if:*

1. *σ induces an r-resilient computational NE;*
2. *There is a noticeable function δ such that for any coalition C with $|C| \leq r$, and any vector of PPT strategies ρ_{-C} that is δ-close to σ_{-C}, any PPT strategy ρ_C, there exists a PPT strategy $\sigma'_C \approx \sigma$ such that $U_i(k, \rho_C, \rho_{-C}) \leq U_i(k, \sigma'_C, \rho_{-C}) + \epsilon(k)$, where $\epsilon(\cdot)$ is negligible.*

Remark 2. Intuitively, the strategy vector (σ_C, σ_{-C}) is stable with respect to trembles if σ_C remains a best response even if P_{-C} plays any PPT strategies other than σ_{-C} with some small but noticeable probability δ.

2.3 Assumptions on the Utility Functions

Following most previous works on this topic, we assume the following properties of the utility functions:

- each player P_i first prefers outcomes in which he outputs the real secret;
- each player P_i secondly prefers outcomes in which the fewest of the other players output the real secret.

As in [7], the expected utility is also assumed to be a polynomial of the security parameter k. We distinguish four cases as follows. For each $i \in \{1, \ldots, n\}$, let $U_i(k)$ (respectively, $U_i^+(k)$) be the minimal (respectively, maximal) payoff of P_i when he outputs the correct secret and let $U_i^-(k)$ be his maximal payoff when P_i does not output s. As usually assumed, we consider: $U_i^+(k) > U_i(k) > U_i^-(k)$ for all $i \in \{1, \ldots, n\}$. As in [7], define

$$U_i^r(k) := \frac{1}{|S|} \cdot U_i^+(k) + (1 - \frac{1}{|S|}) \cdot U_i^-(k)$$

which is the expected utility of a player outputting a random guess for the secret (assuming that the other players abort without any outputs, or with wrong outputs). It is reasonable to assume that $U_i(k) > U_i^r(k)$, since otherwise, players hardly have any incentive to execute the secret reconstruction phase at all. Furthermore, it is still reasonable to assume that the difference between $U_i(k)$ and $U_i^r(k)$ is non-negligible for any $1 \leq i \leq n$, that is, there exists a polynomial $p(\cdot)$ such that for infinitely many k's it holds that:

$$U_i(k) \geq U_i^r(k) + \frac{1}{p(k)}.$$

Note that, this assumption is not restrictive in that without it, it is hard to guarantee the players have enough motivation to execute the share reconstruction phase rather than guess the secret locally, especially in the computational setting, where no player cares about negligible difference in utilities. In this paper, we consider coalitions of at most $t-1$ players. We assume for simplicity that during the whole process of share reconstruction phase, there is at most one coalition which contains a subset of active players and all the players in this coalition share all information they jointly have. Thus, all the players in some coalition are assumed to share a single output.

3 Our Protocol for t-out-of-n Rational Secret Sharing

Our protocol contains two phases: share distribution and secret reconstruction. The first phase is executed by the dealer only while the second phase is designed for all the active players who want to jointly recover the secret without the dealer. Our share distribution phase is similar to the revisited version of the Asmuth-Bloom's non-interactive verifiable SSS [11,3] except with minor but necessary modifications for our needs. The dealer is available only in the initial share distribution phase during which he is assumed to be honest. We assume the existence of synchronous broadcast channels (but non-simultaneous) for all participating players and the presence of private channels between any pair of these players and the dealer.

As said in the previous section, all n players are assumed to be computationally bounded. In the following, let k be a security parameter.

3.1 Initial Share Phase

This is the only phase where the dealer is active. His goal is to distribute s over $\mathcal{P} := \{P_1, \ldots, P_n\}$ using the Asmuth-Bloom SSS with threshold t. As mentioned above, we adopt the modified version of Asmuth-Bloom SSS proposed by Kaya and Selçuk [11] and make further modifications (mainly on the parameters settings) to meet our needs. This initial share phase has two stages.

Remark 3. The value g is the unique integer in $\mathbb{Z}_Q$ satisfying $g_i \equiv g \bmod p_i$, for all $1 \leq i \leq n$. Besides, the order of g in $\mathbb{Z}_{QN}^*$ is at least $\prod_{j=1}^{n} m_j$ and for each $1 \leq i \leq n$, we have:

$$E(y) \bmod p_i = (g^y \bmod QN) \bmod p_i = g^y \bmod p_i = g_i^{y_i} \bmod p_i$$

Hence, during the whole protocol, we use ($E(y)$ mod p_i) as a commitment to y_i, which is perfect binding but is computational hiding. That is, the committer cannot commit himself to two values y_i and y'_i by the same commitment value and, under the assumption that computing discrete logarithm is intractable in $\mathbb{Z}_{p_i}$, no PPT player learns y_i from $E(y)$ mod p_i except with negligible probability in k. This allows players to check the consistency of the received data. Since the dealer is assumed to be honest, $E(y)$ is only used to detect the players' possible malicious behavior during the reconstruction process described in the next section.

Initial Share Phase

1. Parameters Setup

To share a secret s, the dealer chooses $m_0 (> s)$ and publishes it. This value m_0 should also be lower bounded by a value depending on players' utilities and discussed later in this paper.

1. The dealer chooses and publishes a set of pairwise coprime integers $m_1, \ldots, m_n$ of bitlength k such that the following requirements are satisfied:
   (a) $m_0 < m_1 < \ldots < m_n$;
   (b) $\prod_{i=1}^{t} m_i > (n+1) m_0^2 \prod_{i=1}^{t-1} m_{n-i+1}$;
   (c) $p_j = 2m_j + 1$ is prime for any $1 \le j \le n$.
2. For any $1 \le i \le n$, let G_i be a subgroup of $\mathbb{Z}_{p_i}^*$ of order m_i and denote g_i a generator of G_i. Let $Q = \prod_{i=1}^{n} p_i$ and $g = (\sum_{i=1}^{n} g_i \cdot Q'_i \cdot \frac{Q}{p_i})$ mod Q and, where Q'_i is the inverse of $\frac{Q}{p_i}$ in $\mathbb{Z}_{p_i}^*$, for $1 \le i \le n$. The dealer publishes g.
3. The dealer chooses and publishes an RSA modulus N of length at least k whose factorization is unknown to any of the n players [15].

2. Share Distribution

To share a secret $s \in \mathbb{Z}_{m_0}$ among a group of n players $\{P_1, \ldots, P_n\}$, the dealer executes the following steps.

1. He sets $M := \left\lfloor \frac{\prod_{i=1}^{t} m_i}{n+1} \right\rfloor$. He computes $y = s + A_0 \cdot m_0$ for some positive integer A_0 generated randomly subject to the condition that $0 < y < M$, calculates $y_i = y$ mod m_i and finally sends the share y_i to player P_i secretly, for $1 \le i \le n$.
2. He computes $E(y) := g^y$ mod QN and broadcasts $E(y)$.

3.2 Secret Reconstruction Phase

We assume that $m (\ge t)$ players participate in the secret reconstruction phase. For ease of description, we can assume without loss of generality that those players are $P_1, \ldots, P_m$. The reconstruction phase proceeds in a series of iterations, each of which consists of multiple communication rounds among those players. First, we propose two subprotocols to be called upon within the reconstruction phase.

3.2.1 Share Update Phase. This is done by the players participating in the secret reconstruction process, namely, by $P_1, \ldots, P_m$. In this phase, each participating P_i (sorted in index increasing order) plays a similar role to the dealer's (initial share phase) to share a random element $d^{(i)} \in \mathbb{Z}_{m_0}$ and to finally get his "one-time" share for $(s + d^{(1)} + \cdots + d^{(m)}) \bmod m_0$.

In [11], in order to prevent the dealer from distributing inconsistent shares, the range-proof technique proposed from [5] is used to allow the dealer to convince each player that some committed integer lies in a particular interval. This range proof is statistically zero-knowledge in the random-oracle model. Besides, provided that computing discrete logarithm problems is intractable, a cheating dealer can only succeed with negligible probability (in k). We refer to [11,5] for further details.

Here, in order to prevent a player P_i from distributing inconsistent shares for his random chosen $d^{(i)}$, we need to apply this range-proof technique. Throughout this section, we will use $\mathsf{RngPrf}(E(y), M)$ to denote the Cao-Liu's non-interactive range proof that a secret integer y committed with $E(y)$ is in the interval $[0, M)$ [5]. In the following share update phase, we will use $\mathsf{RngPrf}(E(y), M)$ as a black box and we refer to [5] for additional information.

Share Update Phase

1. Each P_i selects a random element $d^{(i)} \in \mathbb{Z}_{m_0}$ uniformly and independently. He computes $y^{(i)} = A_i \cdot m_0 + d^{(i)}$, where A_i is a positive integer chosen randomly conditioned on $0 < y^{(i)} < M$. Then, he computes $y_j^{(i)} = y^{(i)} \bmod m_j$ along with $E(y^{(i)}) := g^{y^{(i)}} \bmod QN$, and he finally sends $y_j^{(i)}$ to player P_j secretly through a secure channel for each $j \neq i$. In addition, P_i broadcasts $E(y^{(i)})$ and $\mathsf{RngPrf}(E(y^{(i)}), M)$.
2. If player P_i only receives partial messages (hereinafter, partial messages including the case of no message at all), then he outputs a random guess of the secret and terminates the protocol. Otherwise, he checks whether $g_i^{y_i^{(j)}} \equiv E(y^{(j)}) \bmod p_i$ and he checks the correctness of $\mathsf{RngPrf}(E(y^j), M)$ for $1 \leq j \neq i \leq m$. If all the checks are successful, then P_i computes $d_i = \sum_{l=1}^{m} y_i^{(l)} \bmod m_i$. Otherwise, he outputs a random guess of the secret and stops the protocol.
   Let $d := d^{(1)} + \cdots + d^{(m)}$. Note that $\{d_1, \ldots, d_m\}$ are the shares for $d \bmod m_0$.
3. Each P_i computes $\widetilde{y_i} := (y_i + d_i) \bmod m_i$ as his "one-time" share for the current iteration. The commitment for $\widetilde{y_i}$ is $E(\widetilde{y_i}) := E(y) \prod_{l=1}^{m} E(y^{(l)}) \bmod p_i$, which can be locally computed by each player.

Proposition 1. *Let $Y := y + y^{(1)} + \ldots + y^{(m)} = (s + d) + (A_0 + \cdots + A_m) \cdot m_0$. Then after the share update phase, $\{\widetilde{y_1}, \ldots, \widetilde{y_m}\}$ are valid shares for $(s + d) \bmod m_0$ as long as all the players follow the protocol honestly. In addition, all the commitments are correctly checked.*

This proposition means that every subset of at least t players uniquely determines $(s + d) \bmod m_0$ (Correctness), while for any subset of $t - 1$ players, every candidate for s or for each $d^{(i)}$ is (approximately) equally likely, and so each candidate for each $(s + d) \bmod m_0$ is (approximately) equally likely (Privacy).

Proposition 2 ([11]). *During the share update phase, any player P_i can not distribute inconsistent shares for $d^{(i)}$ without being detected except with probability negligible in k. In other words, if all checks are successful, then all the shares*

$y_1^{(i)}, \ldots, y_m^{(i)}$ are residues of some integer less than M except with negligible probability which is introduced by the error probability of RngPrf.

Remark 4. Let $T := y^{(1)} + \cdots + y^{(m)}$. Since the "one-time" shares $\{\widetilde{y_1}, \ldots, \widetilde{y_m}\}$ are the shares for $(s + d)$ mod m_0, they are the shares for s if and only if $d \equiv 0$ mod m_0. This is equivalent to $T \equiv 0$ mod m_0. In this sense, the iteration in which $T \equiv 0$ mod m_0 is called a *valid* iteration. It is called an *invalid* iteration otherwise.

Remark 5. The goals of the Share Update Phase are twofold. On one hand, it makes our protocol proceed with several iterations: all except the last one are invalid iterations, which are designed to catch possible cheaters. During the valid iteration, all active players get the real secret. In addition, no one will know in advance whether the current iteration is going to be the last iteration. On the other hand, since during each iteration all the "one-time" shares are revealed, if the current round is invalid, the players should proceed to the next round with totally new shares, which are provided by the share updating phase. Hence, "one-time"shares are shares used only once (i.e. in the current iteration) and they become meaningless in later iterations.

3.2.2 Combiner Phase. In this phase, each player P_i uses the reconstruction algorithm from the Asmuth-Bloom SSS to recover $(s + d)$ mod m_0.

Combiner Phase

1. Let U be a collection of t shares that player P_i chooses in the reconstruction phase and let V be the corresponding collection of the indices of the players to whom those t shares belong. Let M_V denote $\prod_{j \in V} m_j$.
2. Let $M_{V-\{j\}}$ denote $\prod_{\ell \in V, \ell \neq j} m_\ell$ and let $M'_{V,j}$ be the multiplicative inverse of $M_{V-\{j\}}$ in $\mathbb{Z}^*_{m_j}$. Player P_i computes $Y^{(i)} := \sum_{j \in V} \widetilde{y_j} \cdot M'_{V,j} \cdot M_{V-\{j\}}$ mod M_V. Finally, let $S^{(i)} := Y^{(i)}$ mod m_0.

3.2.3 Overview of the Reconstruction Phase. In order for the reader to get an easier understanding of the reconstruction phase, we first give its general view. The full description is in Sect. 3.2.4.

The reconstruction phase proceeds with a sequence of invalid/valid iterations such that the last iteration is valid and each iteration has two stages. During the first stage, players first interact to get their "one-time" shares for $(s+d)$ mod m_0, where $d = d^{(1)} + \cdots + d^{(m)}$ and each $d^{(i)}$ is chosen randomly by P_i. During the second stage, each player P_i is required to open the value $y^{(i)}$ he chose in the first stage. Thus, since $d^{(i)} = y^{(i)}$ mod m_0, the players can jointly identify the status of the current iteration: if d mod $m_0 \neq 0$, then the current iteration is invalid and all the players are asked to restart a new iteration; otherwise, it is valid, the secret s is recovered and the protocol terminates immediately after this iteration.

The iterations have the following properties:

- invalid iteration: no information about s is revealed since all the revealed shares are the shares for $(s+d)$ mod m_0. At the beginning of the subsequent iteration, all the shares are updated which guarantees that the "one-time" shares revealed in the current iteration are useless for the next iteration.
- valid iteration: every player recovers s on the assumption that every participant follows the protocol (which will be demonstrated to be the case since they are rational).

The key in this process is the fact that nobody knows before the opening of the "one-time" shares whether the current iteration will be valid. Furthermore, when a given player realizes that the valid iteration occurs, each other player can compute the secret as well. That is why we do not need simultaneous channels.

3.2.4 Secret Reconstruction Phase. We assume that $m(\geq t)$ players participate in the secret reconstruction. As before, we can assume that they are $P_1, \ldots, P_m$. Our reconstruction protocol proceeds with multiple iterations, each of which contains two stages for each of these m players. It is assumed without lost of generality that in each step, each P_i executes his strategy in index increasing order. For each of these m participants P_i, his strategy σ_i is as follows.

Secret Reconstruction Phase

Stage 1

1. Player P_i executes the share update phase to get his "one-time" share $\widetilde{y_i}$ for the value $(s+d)$ mod m_0, where $d = d^{(1)} + \cdots + d^{(m)}$ and each $d^{(i)}$ is chosen independently and uniformly at random by P_i .
2. Player P_i broadcasts his "one-time" share $\widetilde{y_i}$ obtained at the previous step. If P_i does not receive m shares (including his own), or if he detects that $g_j^{\widetilde{y_j}}$ mod $p_j \neq E(\widetilde{y_j})$ mod p_j for some j, he outputs a random guess of the secret and aborts the protocol abruptly.
3. Otherwise, P_i chooses randomly t data from $\{\widetilde{y_1}, \ldots, \widetilde{y_m}\}$ and executes the Combiner Phase.

The second stage is used to recover T so that player P_i can identify the status (valid/invalid) of the current round since $\{\widetilde{y_1}, \cdots, \widetilde{y_m}\}$ are the shares for s if and only if $T \equiv 0$ mod m_0.

Stage 2

1. Player P_i broadcasts $y^{(i)}$. If he does not receive m messages (including his own), or if he detects that $g^{y^{(j)}}$ mod $QN \neq E(y^{(j)})$ for some j, P_i outputs $S^{(i)}$ he obtains in the Combiner Phase and aborts the whole protocol.
2. Otherwise, P_i computes $T = y^{(1)} + \ldots + y^{(m)}$. If $T \equiv 0$ mod m_0, then he outputs $S^{(i)}$ and stops the whole protocol; Otherwise, P_i goes back to Stage 1 and starts another iteration.

3.3 Security of our Rational SSS

The reconstruction phase is a game amongst the m active players. The strategy profile is denoted $\sigma = (\sigma_1, \ldots, \sigma_m)$ where σ_i is P_i's strategy described in the previous section. Let $U_i^*(k) := \frac{1}{m_0} \cdot U_i^+(k) + (1 - \frac{1}{m_0}) \cdot U_i^r(k)$, $1 \leq i \leq n$. Based on the security requirements of [5], we make the following assumption:

$\mathcal{A}$: The discrete logarithm problem over finite fields is intractable.
The RSA modulus N is hard to factor; the resulting RSA encryption scheme and Schnorr signature is secure.

Theorem 1. *Assuming that $\mathcal{A}$ holds. σ induces a $(t-1)$-resilient computational NE as long as $U_i(k) - U_i^*(k)$ is non-negligible, for $1 \leq i \leq n$.*

Theorem 2. *Assuming that $\mathcal{A}$ holds. σ induces a $(t-1)$-resilient computational strict NE provided that $U_i(k) - U_i^*(k)$ is non-negligible, for $1 \leq i \leq n$.*

Theorem 3. *Assuming that $\mathcal{A}$ holds. σ induces a computational NE that is stable with respect to trembles provided that $U_i(k) - U_i^*(k)$ is non-negligible, for $1 \leq i \leq n$.*

Remark 6. The expected number of iterations of our protocol is m_0. Note that the requirements for m_0 are that $m_0 > \frac{2[U_i^+(k) - U_i^r(k)]}{U_i(k) - U_i^r(k)}$, for $1 \leq i \leq n$. Since all the utility functions are polynomial in k and $U_i(k) - U_i^r(k)$ is assumed to be non-negligible, m_0 can be chosen to be a prime less than some polynomial in k. Since all the computations are based on modular arithmetic, they can be executed in polynomial time. Besides, RngPrf can also be verified in polynomial time. All these considerations imply that our protocol is efficient.

4 Conclusion

In this paper, we presented a new protocol for t-out-of-n rational secret sharing based on the CRT in non-simultaneous channels. Our technique leads to a $(t-1)$-resilient computational strict NE that is stable with respect to trembles while having much smaller share size than the protocol proposed by Fuchsbauer *et al.* [7].

Acknowlegments

The authors would like to thanks the reviewers for their suggestions to improve the quality of this paper. Christophe Tartary's work was funded by the National Natural Science Foundation of China grants 61033001, 61061130540, 61073174 and 61050110147 (International Young Scientists program) as well as the National Basic Research Program of China grants 2007CB807900 and 2007CB807901. Christophe Tartary also acknowledges support from the Danish National Research Foundation and the National Natural Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation (CTIC) within which part of this work was performed. Huaxiong Wang's work was supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In: 25th Annual ACM Symposium on Principles of Distributed Computing (PODC 2006), pp. 53–62. ACM Press, New York (2006)
2. Asharov, G., Lindell, Y.: Utility dependence in correct and fair rational secret sharing. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 559–576. Springer, Heidelberg (2009)
3. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Transactions on Information Theory IT-29(2), 208–210 (1983)
4. Blakley, G.R.: Safeguarding cryptographic keys. In: AFIPS 1979 National Computer Conference, pp. 313–317. AFIPS Press (June 1979)
5. Cao, Z., Liu, L.: Boudot's range-bounded commitment scheme revisited. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 230–238. Springer, Heidelberg (2007)
6. Dov Gordon, S., Katz, J.: Rational secret sharing, revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)
7. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient rational secret sharing in standard communication networks. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 419–436. Springer, Heidelberg (2010)
8. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: 17th Annual ACM Symposium on Theory of Computing (STOC 1985), pp. 291–304. ACM, New York (1985)
9. Halpern, J., Teague, V.: Rational secret sharing and multiparty computation: Extended abstract. In: 36th Annual ACM Symposium on Theory of Computing (STOC 2004), pp. 623–632. ACM Press, New York (2004)
10. Izmalkov, S., Micali, S., Lepinski, M.: Rational secure computation and ideal mechanism design. In: 46th Annual Symposium on the Foundations of Computer Science (FOCS 2005), pp. 585–594. IEEE Computer Society, Los Alamitos (2005)
11. Kaya, K., Selçuk, A.A.: Secret sharing extensions based on the Chinese reminder theorem. Cryptology ePrint Archive, Report 2010/096 (2010), `http://eprint.iacr.org/2010/096`
12. Kol, G., Naor, M.: Games for exchanging information. In: 40th Annual ACM Symposium on Theory of Computing (STOC 2008), pp. 423–432. ACM Press, New York (2008)
13. Micali, S., shelat, a.: Purely rational secret sharing (Extended abstract). In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 54–71. Springer, Heidelberg (2009)
14. Ong, S.J., Parkes, D.C., Rosen, A., Vadhan, S.: Fairness with an honest minority and a rational majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)
15. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
16. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)

A Proof of Theorem 1

By Proposition 1, our protocol is a valid secret sharing scheme. All the active players will be expected to recover the real secret in $\frac{1}{\Pr[d \equiv 0 \bmod m_0]} = m_0$ iterations, as long as they stick to σ.

Now, we prove that σ induces a $(t-1)$-resilient computational NE. Let C be any coalition of size at most $t-1$. Assume that all the players not in C stick to their prescribed strategies. We focus on PPT deviations from some players in C. There are several possible cases: (1) some player P_i in C deviates during the share update phase; (2) some player P_i in C lies about his "one-time" share or only sends partial messages in Stage 1 - Step 2 ; (3) some player P_i in C either opens a fake $y^{(i)}$ or broadcasts nothing in Stage 2.

Suppose (1) happens. There are two possible deviations. **Case 1.** P_i only sends (or broadcasts) partial messages in Stage 2 of share update phase. However, this will be detected and cause the protocol to terminate. In this case, the only profitable thing he can do is to output a random guess of the secret, which will earn him at most $U_i^r(k)$. Obviously, it is a worse outcome to P_i, since $U_i^r(k) < U_i(k)$. Hence, P_i will send all data, fake or real, as required. **Case 2.** P_i distributes inconsistent shares for his randomly chosen $d^{(i)}$ to some player P_j not in C. Under assumption $\mathcal{A}$, no cheating P_i can convince any other P_j to accept $\mathsf{RngPrf}(E(y^{(i)}, M)$ except with negligible probability $\epsilon'(k)$. Once $\mathsf{RngPrf}(E(y^{(i)}, M)$ is rejected, which happens with probability $1-\epsilon'(k)$, the protocol terminates immediately and the best that player P_i can do is to output a random guess of the secret. Thus, the expected utility P_i can get by distributing inconsistent shares is at most $\epsilon'(k) \cdot U_i^+(k) + (1-\epsilon'(k)) \cdot U_i^r(k) = \epsilon'(k) \cdot (U_i^+(k) - U_i^r(k)) + U_i^r(k) < \epsilon(k) + U_i(k)$, where $\epsilon(k) = \epsilon'(k)\,(U_i^+(k) - U_i^r(k))$ is a negligible function in k, since we assumed that $U_1, \ldots, U_n$ were polynomials in k. That is, using this type of deviation, P_i can only increase his payoff by a negligible amount (if at all). Thus, given our computational setting, no rational player P_i is to deviate by distributing inconsistent shares.

Now, we consider the possible deviations in Step 2 of Stage 1. There are two possible cases. **Case 1.** P_i does not broadcast anything at all. **Case 2.** P_i cheats about his "one-time" share. However, either of these deviations will be detected and cause the protocol to terminate. Hence, we do not distinguish between these two cases. If $(d \bmod m_0) = 0$ (i.e., the current iteration is valid which happens with probability $\frac{1}{m_0}$), then all the players in C will output the real secret and hence P_i will get at most $U_i^+(k)$. If $(d \bmod m_0) \neq 0$ (i.e., the current iteration is invalid which happens with probability $1 - \frac{1}{m_0}$), then the best thing P_i can do is to output a random guess of the secret earning at most $U_i^r(k)$. Thus, the expected payoff of P_i with this type of deviation is at most $\frac{1}{m_0} \cdot U_i^+(k) + (1 - \frac{1}{m_0}) \cdot U_i^r(k) = U_i^*(k)$. It is less than $U_i(k)$ by our assumption. Hence, as a rational player, P_i will not deviate in Step 2 of Stage 1.

Finally, we study what happens if some player in C does not broadcast anything at all or broadcast a fake value in Stage 2. Either deviation will be detected and cause the protocol to terminate abruptly. Since we assume players execute every step of the protocol in ascending order, we can assume without loss of generality that $C = \{P_{m-t+2}, \ldots, P_m\}$. Since all the players in C share their information, for any $m-t+2 \leq i \leq m$, after receives the message from the players not in C P_i can first compute $T := y^{(1)} + \cdots + y^{(m)}$ to identify whether the current round is valid or not, then determines what to do in this stage. Note that we have proved that, in the computational and rational setting, any player will execute the reconstruction phase honestly up to the end of Stage 1. Therefore, if the current iteration is valid, each $S^{(j)}$ obtained by P_j in the Combiner Phase is indeed the real secret. In this case, regardless of what P_i will do, each player will output the real secret, which will earn $U_i(k)$ to P_i. On the other hand, if the current round is invalid, no one has recovered the real secret yet and either type of deviations will cause the protocol to terminate abruptly resulting in a payoff at most $U_i^r(k)$ to P_i. Hence, P_i is never better off by this deviations.

B Proof of Theorem 2

Suppose C is any subset of $\{1, \ldots, m\}$ of size at most $t-1$. Let $P_C := \{P_i | i \in C\}$ and $P_{-C} := \{P_i | i \in \{1, \ldots, m\} - C\}$. Since all the players in P_C acts in unison, we can regard P_C as a whole. By Theorem 1, it is sufficient to prove that for any PPT strategy $\rho_C \not\approx \sigma$, there is a positive polynomial $p(\cdot)$ such that for any $i \in C$, $U_i(k, \sigma) \geq U_i(k, \rho_C, \sigma_{-C}) + \frac{1}{p(k)}$ for infinitely many values of k, that is, $U_i(k, \sigma) - U_i(k, \rho_C, \sigma_{-C})$ is positive and non-negligible.

Let $\mathsf{Deviate}$ be the event that P_C deviates from σ_C before he can compute his output, that is, before entering the Stage 2 of the valid iteration. Since $\rho_C \not\approx \sigma$, $\mathrm{Prob}[\mathsf{Deviate}]$ is non-negligible by definition. Now, consider the interaction of ρ_C with σ_{-C}. Let Valid be the event that P_C deviates from σ_C before entering Stage 2 during the valid iteration and let $\mathsf{Invalid}$ be the event that P_C deviates from σ_C during an invalid iteration. Let Caught be the event that P_C is caught cheating. Then, for each $i \in C$, we have:

$$\begin{aligned}
& U_i(k, \rho_C, \sigma_{-C}) \\
\leq\ & U_i^+(k) \cdot \mathrm{Prob}[\mathsf{Valid}] + U_i^+(k) \cdot \mathrm{Prob}[\mathsf{Invalid} \wedge \overline{\mathsf{Caught}}] \\
& + U_i^r(k) \cdot \mathrm{Prob}[\mathsf{Invalid} \wedge \mathsf{Caught}] + U_i(k) \cdot \mathrm{Prob}[\overline{\mathsf{Deviate}}] \\
=\ & U_i^+(k) \cdot (\mathrm{Prob}[\mathsf{Valid}|\mathsf{Deviate}] + \mathrm{Prob}[\overline{\mathsf{Caught}}|\mathsf{Invalid}] \cdot \mathrm{Prob}[\mathsf{Invalid}|\mathsf{Deviate}]) \cdot \mathrm{Prob}[\mathsf{Deviate}] \\
& + U_i^r(k) \cdot \mathrm{Prob}[\mathsf{Caught}|\mathsf{Invalid}] \cdot \mathrm{Prob}[\mathsf{Invalid}|\mathsf{Deviate}] \cdot \mathrm{Prob}[\mathsf{Deviate}] + (1 - \mathrm{Prob}[\mathsf{Deviate}])U_i(k) \\
=\ & U_i^+(k) \cdot \left[\frac{1}{m_0} + \epsilon(k)(1 - \frac{1}{m_0})\right] \cdot \mathrm{Prob}[\mathsf{Deviate}] \\
& + U_i^r(k) \cdot (1 - \epsilon(k)) \cdot (1 - \frac{1}{m_0}) \cdot \mathrm{Prob}[\mathsf{Deviate}] + U_i(k) - U_i(k) \cdot \mathrm{Prob}[\mathsf{Deviate}] \\
=\ & U_i(k) + (U_i^*(k) - U_i(k)) \cdot \mathrm{Prob}[\mathsf{Deviate}] + \eta(k)
\end{aligned}$$

where $\eta(k) = \epsilon(k) \cdot (1 - \frac{1}{m_0}) \cdot (U_i^+(k) - U_i^r(k)) \cdot \mathrm{Prob}[\mathsf{Deviate}]$ is negligible. It follows

$$U_i(k, \sigma) = U_i(k) \geq U_i(k, \rho_C, \sigma_{-C}) + (U_i(k) - U_i^*(k)) \cdot \mathrm{Prob}[\mathsf{Deviate}] - \eta(k).$$

Since both $U_i(k) - U_i^*(k)$ and Prob[$\mathsf{Deviate}$] are positive and non-negligible, $U_i(k,\sigma) - U_i(k,\rho_C,\sigma_{-C})$ is positive and non-negligible, which completes this proof.

Remark 7. In this proof, we actually show that, for any PPT strategy ρ_C, we have:

$$U_i(k,\sigma) = U_i(k) \geq U_i(k,\rho_C,\sigma_{-C}) + (U_i(k) - U_i^*(k)) \cdot \text{Prob}[\mathsf{Deviate}] - \eta(k)$$

where $\eta(\cdot)$ is a negligible function.

C Proof of Theorem 3

This proof is based on [7]. Let δ be a parameter which we will specify at the end of the proof. Note that δ may depend on k. Since we assumed players execute every step of the protocol in an index increasing order, we can assume without loss of generality that $C = \{m-t+2,\ldots,m\}$. It is sufficient to show that for any $i \in C$, any vector of PPT strategies ρ_{-C} that is δ-close to σ_{-C}, and any PPT strategy ρ_C, there exists a PPT strategy $\sigma'_C \approx \sigma$ such that $U_i(k,\rho_C,\rho_{-C}) \leq U_i(k,\sigma'_C,\rho_{-C}) + \epsilon(k)$, where $\epsilon(\cdot)$ is negligible. Let $P_C = \{P_i | i \in C\}$ and $P_{-C} = \{P_i | i \in (\{1,\ldots,m\} - C)\}$. First, we construct a strategy σ'_C for the players in P_C as follows.

```
1. Set Detect:=0.
2. In each iteration:
   (a) Receive the messages from $P_{-C}$ in each possible step. If $P_C$ detects
       that some player $P_j$ in $P_{-C}$ has deviated from $\sigma_j$, set Detect:= 1.
   (b) If Detect= 1, execute the remaining steps according to $\rho_C$; otherwise
       $\sigma_C$.
3. If Detect= 0, determine the output according to $\sigma_C$, otherwise, output
   whatever $\rho_C$ outputs.
```

Observe that when σ'_C interacts with σ_{-C}, Detect is never set to be 1. Hence $\sigma'_C \approx \sigma$ and $U_i(k,\sigma'_C,\sigma_{-C}) = U_i(k,\sigma_C,\sigma_{-C}) = U_i(k)$ for any $i \in C$. Now, we want to show that $U_i(k,\rho_C,\rho_{-C}) \leq U_i(k,\sigma'_C,\rho_{-C}) + \eta(k)$ for any $i \in C$, where $\eta(\cdot)$ is negligible. Let $\widetilde{\rho_{-C}}$ denote the residual strategy of ρ_{-C}. In an interaction where P_C follows strategy ρ_C, let $\mathsf{Detected}$ be the event that P_C is detected deviating from σ_C before entering stage 2 of the valid iteration while no player in P_{-C} is detected cheating so far. Also, let $\text{Prob}_{\mathsf{Detected}}(\alpha)$ be the probability of $\mathsf{Detected}$ when P_{-C} follows strategy α. Since no player in P_{-C} will be detected cheating when P_{-C} execute σ_{-C}, $\text{Prob}_{\mathsf{Detect}ed}(\sigma_{-C})$ equals the probability of P_C being detected deviating from σ_C before entering Stage 2 of the valid iteration.

Claim 1. Prob[$\mathsf{Deviate}$]=$\text{Prob}_{\mathsf{Detected}}(\sigma_{-C}) + \epsilon(k) \cdot \text{Prob}[\mathsf{Deviate}]$, for some negligible function ϵ.

Claim 2. For any $i \in C$,

$$U_i(k, \rho_C, \widetilde{\rho_{-C}}) - U_i(k, \sigma'_C, \widetilde{\rho_{-C}}) \leq \text{Prob}_{\mathsf{Detected}}(\widetilde{\rho_{-C}}) \cdot (U_i^+(k) - U_i^r(k)) + \epsilon(k),$$

where $\epsilon(\cdot)$ is negligible.

Claim 3. $\text{Prob}_{\mathsf{Detected}}(\widetilde{\rho_{-C}}) \leq \text{Prob}_{\mathsf{Detected}}(\sigma_{-C}) + \epsilon(k)$ for some $\epsilon(\cdot)$ negligible.

By Remark 7, we know that for any PPT strategy ρ_C,

$$U_i(k, \sigma) = U_i(k) \geq U_i(k, \rho_C, \sigma_{-C}) + (U_i(k) - U_i^*(k)) \cdot \text{Prob}[\mathsf{Deviate}] - \eta(k).$$

where $\eta(\cdot)$ is a negligible function. Now, we get:

$$\begin{aligned} U_i(k, \rho_C, \rho_{-C}) &= (1-\delta) \cdot U_i(k, \rho_C, \sigma_{-C}) + \delta \cdot U_i(k, \rho_C, \widetilde{\rho_{-C}}) \\ &\leq (1-\delta) \cdot [U_i(k) + (U_i^*(k) - U_i(k)) \cdot \text{Prob}[\mathsf{Deviate}] + \eta(k)] \\ &\quad + \delta \cdot U_i(k, \rho_C, \widetilde{\rho_{-C}}) \end{aligned}$$

Also

$$\begin{aligned} U_i(k, \sigma'_C, \rho_{-C}) &= (1-\delta) \cdot U_i(k, \sigma'_C, \sigma_{-C}) + \delta \cdot U_i(k, \sigma'_C, \widetilde{\rho_{-C}}) \\ &= (1-\delta) \cdot U_i(k) + \delta \cdot U_i(k, \sigma'_C, \widetilde{\rho_{-C}}) \end{aligned}$$

It follows:

$$\begin{aligned} & U_i(k, \rho_C, \rho_{-C}) - U_i(k, \sigma'_C, \rho_{-C}) \\ \leq\ & (1-\delta) \cdot (U_i^*(k) - U_i(k)) \cdot \text{Prob}[\mathsf{Deviate}] + \delta \cdot [U_i(k, \rho_i, \widetilde{\rho_{-C}}) - U_i(k, \sigma'_C, \widetilde{\rho_{-C}}] + \eta(k) \\ \overset{\text{by Claim 2}}{\leq}\ & (1-\delta) \cdot (U_i^*(k) - U_i(k)) \cdot \text{Prob}[\mathsf{Deviate}] \\ & + \delta \cdot \text{Prob}_{\mathsf{Detected}}(\widetilde{\rho_{-C}}) \cdot (U_i^+(k) - U_i^r(k)) + \delta \cdot \epsilon(k) + \eta(k) \\ \overset{\text{by Claim 1}}{=}\ & (1-\delta) \cdot (U_i^*(k) - U_i(k)) \cdot (\text{Prob}_{\mathsf{Detected}}(\sigma_{-C}) + \epsilon'(k) \cdot \text{Prob}[\mathsf{Deviate}]) \\ & + \delta \cdot (U_i^+(k) - U_i^r(k)) \cdot \text{Prob}_{\mathsf{Detected}}(\widetilde{\rho_{-C}}) + \delta \cdot \epsilon(k) + \eta(k) \\ \overset{\text{by Claim 3}}{\leq}\ & (1-\delta) \cdot (U_i^*(k) - U_i(k)) \cdot \text{Prob}_{\mathsf{Detected}}(\widetilde{\rho_{-C}}) \\ & + \delta \cdot (U_i^+(k) - U_i^r(k)) \cdot \text{Prob}_{\mathsf{Detected}}(\widetilde{\rho_{-C}}) + \eta'(k) \end{aligned}$$

where $\eta'(\cdot)$ is some negligible function. Hence, there exists $\delta > 0$ (may depend on k) such that the above expression is negligible in k for each $i \in C$.