A Verifiable 1-out-of-n Distributed Oblivious Transfer Protocol[*]

Christian L. F. Corniaux and Hossein Ghodosi

James Cook University,
Townsville QLD 4811,
Australia

chris.corniaux@my.jcu.edu.au, hossein.ghodosi@jcu.edu.au

February 7, 2013

Abstract

In the various 1-out-of-n distributed oblivious transfer protocols (DOT) designed in an unconditionally secure environment, a receiver contacts k out of m servers to obtain one of the n secrets held by a sender. After a protocol has been executed, the sender has no information on the choice of the receiver and the receiver has no information on the secrets she did not obtain. Likewise, a coalition of $k-1$ servers is unable to infer any information, neither on the sender's secrets, nor on the receiver's choice.

These protocols are based on a semi-honest model: no mechanism prevents a group of malicious servers from disrupting the protocol such that the secret obtained by the receiver does not correspond to the chosen secret. Actually, to verify the information transmitted by the servers seems to require some properties difficult to reconcile: on one hand the receiver has to collect more information from the servers to discard the incorrect data generated by the malicious servers; on the other hand, if the receiver is allowed to gather more information from the servers, the sender's security may be compromised.

We study the first unconditionally secure DOT protocol in the presence of an active adversary who may corrupt up to $k-1$ servers. In addition to the active adversary, we also assume that the sender may (passively) corrupt up to $k-1$ servers to learn the choice of the receiver. Similarly, the receiver may (passively) corrupt up to $k-1$ servers to learn more than the chosen secret. However, we assume that the sender, receiver, and active adversary do not collaborate with each other. Our DOT protocol allows the receiver to contact $4k-3$ servers to obtain one secret, while the required security is maintained.

Keywords: Cryptographic Protocol, Privacy and Security, Distributed Oblivious Transfer, Verifiable Oblivious Transfer

1 Introduction

In the unconditionally secure distributed oblivious transfer (DOT) schemes presented in [14, 3, 15, 4], a sender $\mathcal{S}$ holds n secrets and a receiver $\mathcal{R}$ wishes to obtain one of them. The model encompasses a distributed environment including m servers. The sender distributes shares of the secrets to the servers and does not intervene in the rest of the protocol. The receiver selects the index of a secret, sends shares of this index to k servers and receives back k shares allowing her

[*]The ongoing work related to this study was presented in a poster session at ACISP 2011

to reconstruct the chosen secret. The security of these protocols may be assessed thanks to the following four security conditions defined by Blundo, D'Arco, De Santis and Stinson [3, 4]:

C_1. Correctness – The receiver is able to determine the chosen secret once she receives information from the k contacted servers.

C_2. Receiver's privacy – A coalition of up to $k-1$ servers cannot obtain any information on the choice of the receiver.

C_3. Sender's privacy with respect to $k-1$ servers and the receiver – A coalition of up to $k-1$ servers with the receiver does not obtain any information about the secrets before the protocol is executed.

C_4. Sender's privacy with respect to a "greedy" receiver – Given the transcript of the interaction with k servers, a coalition of up to $k-1$ dishonest servers and the receiver does not obtain any information about secrets which were not chosen by the receiver.

In these protocols, security condition C_1 is satisfied in a semi-honest model; No mechanism prevents a set of up to $k-1$ malicious servers from disrupting the protocol such that the secret obtained by the receiver does not correspond to the chosen secret. Such a mechanism could allow (1) the servers to check that the requests sent by the receiver are consistent and (2) the receiver to collect redundant shares to detect and discard inconsistent shares returned by the malicious servers.

We introduce this kind of mechanism in one of the (k, m)-DOT-$\binom{n}{1}$ protocols presented by Blundo et al. [3, 4]. In this context, condition C_1 is extended to condition C'_1:

C'_1. Correctness – If the receiver is not detected as cheating, she is able to determine the chosen secret once she receives information from the contacted servers, in spite of $k-1$ malicious servers.

Our protocol guarantees security conditions C'_1, C_2 and C_3, despite the presence of up to $k-1$ malicious servers among the m servers participating to the protocol. Blundo et al. have proven that security condition C_4 cannot be satisfied by one-round DOT protocols. Indeed, when condition C_2 is satisfied, any subset of $k-1$ shares selected from the k shares collected by the receiver during the execution of the protocol, cannot give any information on the receiver's choice. Having selected a subset of $k-1$ shares, the receiver is able to build a request to collect a k^{th} share – from a corrupted server – to obtain any secret.

This paper is organised as follows. The next section shortly describes Blundo et al.'s protocol [4] and investigates verifiable secret sharing schemes. In Sect. 3 we introduce a few notations and definitions used in the rest of the paper. Then, in Sect. 4, we describe our model and give an overview of our verifiable DOT protocol. The main components of our protocol are detailed in Sect. 5. Sect. 6 is devoted to the detailed description of the protocol. Finally, in Sect. 7, we analyse the security of the protocol.

2 Related Works

Although there have been a few DOT protocols studied in the past 15 years, e.g. [14], [3, 15, 22, 23, 4], the verifiability of distributed shares in such protocols was rarely tackled. The main contribution to the subject was made by Zhong and Yang [22, 23], but the setting of their proposed scheme is conditionally secure (difficulty to compute a discrete logarithm).

We present the first unconditionally secure verifiable DOT, adapted from the main component of the DOT protocol introduced by Blundo et al. [3, 4]. This component is described hereafter.

2.1 Blundo et al.'s Distributed Oblivious Transfer

The basic principles underlying DOT protocols are conceptually similar. In the original DOT protocol [14] introduced by Naor and Pinkas, as well as in its generalization [3, 4] presented by Blundo et al., a sender distributes some information amongst m servers so that, by contacting k servers, a receiver is able to learn only one of the secrets held by the sender. A simplified overview of the (k, m)-DOT-$\binom{n}{1}$ presented in [4] may be described as follows (operations are executed in a finite field $\mathbb{F}_p$, where p is a prime number):

1. The sender, who holds n secrets $\omega_0, \ldots, \omega_{n-1}$ generates a sparse n-variate polynomial function Q defined by

$$Q\left(x, y_1, \ldots, y_{n-1}\right) = \omega_0 + \sum_{i=1}^{k-1} a_i x^i + \sum_{i=1}^{n-1} \left(\omega_i - \omega_0\right) \times y_i,$$

   where the coefficients a_i $(1 \leq i \leq k-1)$ are numbers randomly selected in $\mathbb{F}_p$. We note that $\omega_0 = Q\left(0, \ldots, 0\right)$ and, for $\ell \in \{1, \ldots, n-1\}$, $\omega_\ell = Q\left(0, \ldots, 0, 1, 0, \ldots, 0\right)$, where the number 1 is in position $\ell + 1$.

2. Then, to each server S_j $(1 \leq j \leq m)$, the sender transmits the $(n-1)$-variate polynomial function F_j defined by

$$F_j\left(y_1, \ldots, y_{n-1}\right) = Q\left(j, y_1, \ldots, y_{n-1}\right) \quad .$$

3. In the transfer phase, the receiver chooses the identifier ℓ of one secret and generates univariate polynomial functions Z_i $(1 \leq i \leq n-1)$ of degree at most $k-1$ such that $(Z_1(0), \ldots, Z_{n-1}(0))$ is an $(n-1)$-tuple of zeros if the receiver is interested in ω_0 (i.e., $\ell = 0$), or an $(n-1)$-tuple of zeros and a single one in position ℓ if the receiver is interested in ω_ℓ (where $\ell \in \{1, \ldots, n-1\}$).

4. Then, the receiver selects a subset $\mathcal{I}_k \subset \{1, \ldots, m\}$ of k indices and sends to each server S_i $(i \in \mathcal{I}_k)$ a request $(i, Z_1(i), \ldots, Z_{n-1}(i))$. When a server S_i receives such a request, it replies with the share $F_i\left(Z_1(i), \ldots, Z_{n-1}(i)\right)$.

5. After receiving k responses, the receiver interpolates a univariate polynomial R from the k points $(i, F_i\left(Z_1(i), \ldots, Z_{n-1}(i)\right))$ and calculates the chosen secret: $\omega_\ell = R\left(0\right)$.

Moreover, the secrets are masked and the protocol is executed twice: the first instance allows the receiver to obtain a masked secret and the second instance allows her to obtain the corresponding mask.

2.2 Verifiable Secret Sharing Schemes

A component common to all unconditionally secure DOT protocols is the threshold secret sharing scheme introduced by Shamir [19] in 1979. In this scheme, the dealer who shares a secret is honest: the m pieces he generates are built in compliance with the protocol and so, the k pieces used by the players to reconstruct the secret are genuine. In 1985, Chor, Goldwasser, Micali and Awerbuch [6] assumed that the dealer could be cheating and transmit some invalid shares to some players. They introduced the concept of verifiable secret sharing (VSS) to detect any deviation of the dealer from the sharing protocol. Moreover, during the reconstruction phase, some malicious players could provide honest players with incorrect shares to make honest players reconstruct an incorrect secret while they, the dishonest players, would reconstruct the original secret. Tompa and Woll [21] studied the particular problem of secret sharing in presence of an honest dealer but also in the presence of dishonest players attempting to cheat during the reconstruction of the secret.

Following Chor et al.'s scheme, there have been a considerable research on VSS schemes. It is clear that if a combiner – or a group of players – wants to check the validity of a reconstructed secret, she needs to collect in addition to k shares, some verification information. Basically, this verification information may consist of additional shares (redundancy technique) or longer shares containing verification information (check vectors technique).

The first unconditionally secure VSSs were introduced by Ben-Or, Goldwasser and Wigderson [1] and by Chaum, Crépeau and Damgård [5]. However, the correctness of the VSS presented in [5] is not guaranteed: the reconstructed secret may be incorrect with a very small probability. A similar weakness can be found in [17, 16] and [7].

The unconditionally secure VSS introduced by Ben-Or et al. [1] tolerates $\frac{n}{3}$ cheaters – n being the number of players – including the dealer, and is correct without probability error. This scheme was improved by Cramer, Damgård and Maurer [8] and a simpler version was proposed by Desmedt, Kurosawa and Van Le [10]. However, in the original scheme and its variants, the dealer has to make public the shares of complaining players. So, cheating players may force the dealer to reveal some shares and consequently the threshold k cannot be guaranteed. In this scenario, a coalition of $k-1$ corrupt servers (See security conditions C_3 and C_4) could obtain additional shares to the shares it holds, and consequently breach the security of the protocol.

In [20], Stinson and Wei introduced an unconditionally secure VSS in which the shares of the identified cheaters are not made public, but are simply ignored during the reconstruction of the secret. Therefore, the threshold k of the underlying secret sharing scheme is not modified during the execution of the protocol. The scheme is correct without error probability and tolerates up to $t = \lfloor \frac{n}{4} \rfloor$ cheaters. Gennaro, Ishai, Kushilevitz and Rabin proposed a VSS [13] similar to Stinson et al.'s scheme, tolerating also up to $t = \lfloor \frac{n}{4} \rfloor$ cheaters and with no share publicly disclosed. D'Arco and Stinson [9] improved this VSS scheme in 2002 using a feature introduced in [13] to detect inconsistencies.

However, in [20] and in [9], the detection of cheaters is based on the determination of a *maximum clique* in a graph, which is a classical NP-complete problem; Thus, the complexity of these two VSS is not polynomial. Inversely, in [13], the detection of cheaters is based on the determination of a *maximal matching* in a graph, which is a problem that can be solved in a polynomial time [11]. For these reasons, we have chosen to replace the secret sharing scheme component used in [4] with Gennaro et al.'s VSS.

3 Preliminaries

Throughout this paper, operations are executed in a finite field $\mathbb{K} = \mathbb{F}_p$ (p prime). We assume that $p \geq \max(n, \omega_0, \ldots, \omega_{n-1}, m)$, where n is the number of secrets, m is the number of servers involved in the protocol, and $\omega_0, \ldots, \omega_{n-1}$ are the n secrets in the system. In addition, by an abuse of language, a polynomial and its corresponding polynomial function will not be differentiated.

Definition 1. If $(\mathbb{K}[X], +, \times)$ is the ring of polynomials over $\mathbb{K}$ and $(\mathbb{K}_k[X], +)$ the additive group of polynomials of degree at most k over $\mathbb{K}$, we say that a polynomial $F = \sum_{i=0}^{k} f_i X^i$ of

$\mathbb{K}_k[X]$ is *quasi-random*, if the coefficients f_i $(1 \leq i \leq k)$ are randomly selected in $\mathbb{K}$ and the constant term $f_0 \in \mathbb{K}$ has a predefined value.

This definition is extended to bivariate polynomials. If $(\mathbb{K}[X,Y],+,\times)$ is the ring of bivariate polynomials over $\mathbb{K}$ and $(\mathbb{K}_k[X,Y],+)$ the additive group of polynomials of degree at most k in X and Y over $\mathbb{K}$, we say that a polynomial $F = \sum_{i=0}^{k}\sum_{j=0}^{k} f_{i,j}X^iY^j$ of $\mathbb{K}_k[X,Y]$ is *quasi-random*, if the coefficients $f_{i,j}$ $(0 \leq i,j \leq k, (i,j) \neq (0,0))$ are randomly selected in $\mathbb{K}$ and the constant term $f_{0,0}$ has a predefined value.

Definition 2. In a $(k,\ m)$-threshold secret sharing scheme involving a dealer $\mathcal{D}$, m players $\mathcal{P}_1,\ldots,\mathcal{P}_m$ and a combiner $\mathcal{C}$, if the same polynomial of $\mathbb{K}_{k-1}[X]$ is interpolated from any set of k shares selected from the m shares distributed by the dealer $\mathcal{D}$ to players $\mathcal{P}_1,\ldots,\mathcal{P}_m$, then the m shares are said *k-consistent*.

Definition 3. In a $(k,\ m)$-threshold secret sharing scheme involving a dealer $\mathcal{D}$, m players $\mathcal{P}_1,\ldots,\mathcal{P}_m$ and a combiner $\mathcal{C}$, if a set of ℓ shares $(k \leq \ell \leq m)$, possibly including incorrect shares, allows $\mathcal{C}$ to calculate the dealer's secret s, then the ℓ shares are said *k-resilient*.

Notations:
The Kronecker's symbol, δ_i^j, is equal to 0 if $i \neq j$ and equal to 1 if $i = j$.
By $\alpha \in_R \mathbb{K}$, we mean that α is chosen randomly from all possible elements of $\mathbb{K}$.
If $\boldsymbol{v} = (f_1,\ldots,f_n)$ is a vector of polynomials and i an element of $\mathbb{K}$, we denote $\boldsymbol{v}(i)$ the vector $(f_1(i),\ldots,f_n(i))$.

4 Our Model

The setting of our model encompasses a sender $\mathcal{S}$ who holds n secrets $\omega_0,\ldots,\omega_{n-1}$ $(n > 1)$, a receiver $\mathcal{R}$ who wishes to obtain a secret ω_σ $(\sigma \in \{0,\ldots,n-1\})$, and m servers $S_1,\ldots,S_m$. In addition to these parties, we also need to take into account an adversary whose characteristics are defined below.

To detect cheating, the servers must be able to communicate with each other and also to have access to a broadcast channel.

Like other DOT protocols, our protocol is composed of two parts: a set-up part and an oblivious transfer part. During the set-up part, the sender generates a sharing polynomial for the n secrets he holds and distributes shares of this sharing polynomial to the m servers. The sender does not intervene in the rest of the protocol. During the oblivious transfer part, the receiver has to contact $c \geq 4k-3$ servers $(1 < c < m)$ to collect enough shares to construct ω_σ.

4.1 Adversary Model

Three parties may try to breach the security of our protocol:

- The sender $\mathcal{S}$, with possibly a coalition of up to $k-1$ corrupt servers, plotting against the receiver to obtain the receiver's choice σ. Servers of the coalition make any information they hold available to the sender. In addition, $\mathcal{S}$ distributes correct shares of the secrets he holds to the servers.

- The receiver $\mathcal{R}$, colluding with up to $k-1$ corrupted servers to obtain information not only about the chosen secret ω_σ, but about other secrets too. Here too, the servers of the coalition share their data with $\mathcal{R}$.

- An active adversary, with the help of a group of up to $k-1$ malicious servers, who intends to disrupt the protocol such that the secret obtained by the receiver does not correspond to the chosen secret. In particular, when they are requested to provide a share, these malicious servers may not reply or replace the requested share with any value, designated in this case as an *incorrect* share.

We assume that both the sender $\mathcal{S}$ and the receiver $\mathcal{R}$ wish to complete the protocol to allow $\mathcal{R}$ to obtain the chosen secret. The adversary collaborates neither with the sender $\mathcal{S}$ nor with the receiver $\mathcal{R}$. Therefore, we assume that the set of malicious servers, the set of servers colluding with $\mathcal{S}$ and the set of servers colluding with $\mathcal{R}$ are disjoint.

In this paper, we consider static parties only; the sets of malicious and corrupted servers are in place before the protocol is executed and their contents do not change during the execution of the protocol.

In addition, along the protocol, some servers may be disqualified. We assume that a mechanism prevents the disqualified servers from keeping on participating in the protocol.

4.2 Communication Model

Our protocol requires the availability of private communication channels between the sender and the servers, the receiver and the servers and among the servers. It also requires a broadcast channel, allowing all participants to receive simultaneously information sent by one participant through this channel.

We assume that these communication channels are secure, i.e., any party is unable to eavesdrop on them and they guarantee that communications cannot be tampered with.

4.3 Principle of the Protocol

The protocol introduced by Blundo et al. (See Sect. 2.1) is adapted with the following two modifications:

1. The receiver is allowed to contact more than k servers. However, to prevent the receiver from obtaining more than one secret (See the impossibility result with security condition C_4 in Sect. 1), she must be kept from asking information related to different secrets. Distributing the shares of her secret inputs to $c \geq 4k-3$ servers, thanks to a VSS scheme, enables the servers to verify that the receiver did not cheat, to disqualify $\ell \leq k-1$ servers with incorrect shares and to prepare at least $c-\ell$ k-consistent shares.

2. The receiver collects at least $k+2t$ shares, including $t \leq k-1$ incorrect shares. An error-decoding code decoding scheme allows her to interpolate a unique sharing polynomial $R \in \mathbb{K}_{k-1}[X]$ and to calculate the chosen secret $R(0)$.

To guarantee the security of the sender and the privacy of the receiver, the sender and the receiver send shares of their privates values to the servers. That is, the sender generates a sharing n-variate polynomial from the the n secrets he holds and distributes one share to each of the m servers. Actually, the shares are $(n-1)$-variate polynomials. To obtain a secret ω_σ, the receiver selects at least $4k-3$ servers, and for each secret input δ_σ^s $(1 \leq s \leq n-1)$ distributes shares generated by Gennaro et al.'s VSS to these servers. For each secret input, the VSS allows the majority of honest servers to identify servers with incorrect shares and to disqualify some of them. If the overall number of disqualified servers is greater than k, the receiver has cheated and the protocol stops. Otherwise, servers with inconsistent shares are able to correct them, thanks to an error-correcting codes decoding scheme. Every server not disqualified returns to the receiver the evaluation of the $(n-1)$-variate polynomial received from the sender at the vector of $n-1$ shares transmitted by the receiver. Using the same error-correcting codes decoding scheme as the servers, the receiver then is able to determine a sharing polynomial related to the chosen secret and to calculate this secret.

5 Components of the System

Our protocol is mainly based on two components.

The first one is an error-correcting codes decoding scheme [18, 12]. With this scheme, a participant who has collected $k+2t$ shares generated by a sharing polynomial of $\mathbb{K}_{k-1}[X]$ is able to determine the dealer's secret, even if t out of the $k+2t$ shares are incorrect.

The second component is the VSS introduced by Gennaro et al. [13]. This scheme allows $4k-3$ players which have received shares generated from a sharing polynomial of $\mathbb{K}_{k-1}[X]$ to detect and disqualify up to $k-1$ cheating participants (dealer or players). The scheme also enables the players to prepare k-consistent shares.

5.1 Error-Correcting Codes Decoding Scheme

Let $\mathcal{D}$ be an honest dealer holding a secret s, $\mathcal{P}_i$ $(i=1,\ldots,m)$ a player and $\mathcal{C}$ a combiner wishing to obtain s. We assume that $\mathcal{D}$ uses for example Shamir's $(k,\, m)$-threshold scheme to generate m shares s_i $(1 \leq i \leq m)$ from a polynomial f and distributes the share s_i to the player $\mathcal{P}_i$. Thus, if $\mathcal{C}$ collects k shares from the players, she can reconstruct s, but if $\mathcal{C}$ collects $k-1$ or less shares, she does not obtain any information on s. We also assume that up to t players may cheat; a cheating player $\mathcal{P}_c$, when requested by $\mathcal{C}$ to provide the share s_c received from $\mathcal{D}$, may transmit to $\mathcal{C}$ a value $s'_c \neq s_c$.

If $\mathcal{C}$ contacts k players, she will collect k shares (correct or incorrect) and will be able to interpolate a polynomial f' of degree at most $k-1$ and to calculate $s' = f'(0)$. If only one share is incorrect, $s' \neq s$. So, we consider that $\mathcal{C}$ is allowed to contact ℓ players $(k \leq \ell \leq m)$ to collect some redundant information.

If the ℓ shares are not k-consistent, we propose a protocol hereafter such that if $\ell \geq k+2t$, then $\mathcal{C}$ is able to reconstruct the original secret, in spite of up to t incorrect shares transmitted by cheaters. This protocol is adapted from the Reed-Solomon correcting error technique [18] which allows a decoder, given a list of $k+2t$ codes including up to t incorrect codes, to detect that some codes are incorrect, to locate them and to correct them.

Set-up Phase

Each player $\mathcal{P}_i$ $(1 \leq i \leq m)$ is allocated a distinct number $\alpha_i \in \mathbb{K}$. The values $\alpha_1,\ldots,\alpha_m$ are public. The sharing polynomial $f = \sum_{i=0}^{k-1} a_i X^i$, where $a_0 = s$ and $a_i \in_R \mathbb{K}$ $(1 \leq i \leq k-1)$, is presented under the form of a word $a = (a_0,\ldots,a_{k-1})$ encoded as the codeword $D = (D_1,\ldots,D_m)$, where $D_i = f(\alpha_i)$ $(1 \leq i \leq m)$.

The dealer $\mathcal{D}$ distributes $(\alpha_i,\, D_i)$ to the player $\mathcal{P}_i$ $(1 \leq i \leq m)$.

Reconstruction Phase

The combiner $\mathcal{C}$ collects ℓ shares $(k \leq \ell \leq m)$, without loss of generality from players $\mathcal{P}_1,\ldots,\mathcal{P}_\ell$, and checks that the collected shares are k-consistent. To perform this task, $\mathcal{C}$ applies Lagrange interpolation formula on the first k collected shares $(\alpha_i,\, D_i)$ $(1 \leq i \leq k)$ and obtains a polynomial f of degree at most $k-1$. For all remaining shares $(\alpha_j,\, D_j)$ $(k+1 \leq j \leq \ell)$, $\mathcal{C}$ checks that $f(\alpha_j) = D_j$. If the $\ell - k$ equalities are satisfied, the shares are k-consistent and $\mathcal{C}$ calculates $s = f(0)$.

If incorrect shares were detected, they could be identified and corrected thanks to algorithms like the Berlekamp-Massey algorithm [2] introduced by Berlekamp. However, if the main objective of error-correcting codes is to restore original codes from corrupted codes, our goal is to reconstruct the polynomial f which was used to generates the code so as to calculate $s = f(0)$. In other words, the combiner $\mathcal{C}$ does not need to identify the incorrect shares, but needs to interpolate f from the received shares. This is why a Reed-Solomon decoding algorithm like the algorithm introduced by Gao [12] is preferred. This algorithm, described hereafter, allows $\mathcal{C}$ to reconstruct f, in spite of up to $t \leq \frac{\ell-k}{2}$ incorrect shares.

First, the combiner $\mathcal{C}$ generates two polynomials g_0 and g_1 such that:

- $g_0 = \prod_{i=1}^{\ell} (x - \alpha_i)$ and
- g_1 is interpolated from the received shares $D_1, \dots, D_\ell$. Thus, g_1 is the unique polynomial of $\mathbb{K}_{\ell-1}[X]$ such that $g_1(\alpha_i) = D_i$ for $1 \leq i \leq \ell$.

Then, $\mathcal{C}$ calculates a partial GCD polynomial g between g_0 and g_1. This polynomial g is determined using the extended Euclidean algorithm. More precisely, $\mathcal{C}$ determines three polynomials u, v and g verifying $ug_0 + vg_1 = g$ and the following criterion. At each step i of the algorithm, $\mathcal{C}$ obtains u_i, v_i and r_i such that $u_i g_0 + v_i g_1 = r_i$. At each step $i+1$ of the iteration, the degree of r_{i+1} is smaller than the degree of r_i obtained in the previous step. As soon as $\deg r_i < \frac{\ell+k}{2}$, the iteration process is stopped and $v = v_i$ and $g = r_i$. Then, $\mathcal{C}$ determines the polynomial $\frac{g}{v}$ which is the polynomial f generated by $\mathcal{D}$ in the set-up phase, and consequently can calculate $s = f(0)$.

5.2 Verifiable Secret Sharing Scheme

The setting of Gennaro et al.'s VSS scheme [13] encompasses a dealer $\mathcal{D}$ who holds a secret ω, m players $\mathcal{P}_1, \dots, \mathcal{P}_m$, each of them receiving a $(k,\, m)$-threshold share of ω and a combiner $\mathcal{C}$ collecting the shares from ℓ $(1 < \ell < m)$ players $\mathcal{P}_1, \dots, \mathcal{P}_\ell$. Up to $t < k$ participants (dealer or players) may cheat during the execution of the protocol. In particular, cheating players may provide incorrect shares to $\mathcal{C}$. The protocol guarantees that, if $\ell \geq 4t+1$, $\mathcal{C}$ is able to verify the k-consistency of the shares distributed by $\mathcal{D}$ and also to collect a set of k-resilient shares if $\mathcal{D}$ is not detected as cheating.

The outline of the protocol is given hereafter.

First, $\mathcal{D}$ generates a quasi-random bivariate polynomial F in $\mathbb{K}_{k-1}[X, Y]$. We denote $f_i(x)$ the polynomial $F(x, i)$ where $i \in \{1, \dots, m\}$. Similarly, we denote $g_i(y)$ the polynomial $F(i, y)$ where $i \in \{1, \dots, m\}$. The dealer $\mathcal{D}$ distributes to player $\mathcal{P}_i$ $(1 \leq i \leq m)$ the polynomials f_i and g_i.

Second, each player $\mathcal{P}_i$ $(1 \leq i \leq m)$ sends to the player $\mathcal{P}_j$ $(1 \leq j \leq m)$ a number $r_{i,j} \in_R \mathbb{K}$ and then broadcasts $f_i(j) + r_{i,j}$ and $g_i(j) + r_{j,i}$. From the broadcast values, each player builds the same set of lists $\{L_1, \dots, L_m\}$ where L_i contains $\mathcal{P}_j$ if $f_i(j) + r_{i,j} \neq g_j(i) + r_{i,j}$.

Third, each player transforms the lists $L_1, \dots, L_m$ into a graph $\mathcal{G} = (V, E)$ where $V = \{\mathcal{P}_1, \dots, \mathcal{P}_m\}$ and $(\mathcal{P}_i, \mathcal{P}_j) \in E$ if $\mathcal{P}_i \notin L_j$ and $\mathcal{P}_j \notin L_i$. The following step allows the players to build three sets H, D and C. The set H contains players whose broadcast information is consistent with the broadcast information of all other players of H, D is the group of players whose broadcast information is consistent with the broadcast information of at least $2t+1$ players in H, and C is the set of disqualified players. Players in D and C may have cheated or they may be honest but have received forged shares from the dealer.

Remark. It is important to note that the players in $H \cup D$ are able to calculate their correct shares, but may be cheaters, whereas players in C do not hold enough correct information to reconstruct the shares they should hold.

Applying for example the algorithm described in Appendix A, $\mathcal{P}_i$ determines a maximal matching M_{al} of $\overline{\mathcal{G}}$ and builds the set of players $\overline{H} = \bigcup\{\mathcal{P}_i, \mathcal{P}_j\}$ where $(\mathcal{P}_i, \mathcal{P}_j) \in M_{\text{al}}$ and the set $H = \{\mathcal{P}_1, \ldots, \mathcal{P}_m\} \setminus \overline{H}$. Then, $\mathcal{P}_i$ builds the set of players D, thanks to the following algorithm: for each $\mathcal{P}_u \notin H$, $\mathcal{P}_i$ checks the number n_u of pairs $(\mathcal{P}_u, \mathcal{P}_v)$ in E. If $n_u \geq 2t+1$, then $\mathcal{P}_u$ is added to D. Otherwise, $\mathcal{P}_u$ is disqualified and added to the set C of cheaters.

The last step consists for $\mathcal{P}_i$ in assessing $|H| + |D|$: if $|H| + |D| < 3t+1$, then $\mathcal{D}$ has cheated. Otherwise, each player $\mathcal{P}_i$ calculates $f_i(0)$.

At the end of the protocol, if $\mathcal{D}$ was not detected as a cheater, $|H| + |D| \geq 3t+1$ players, including up to t corrupted players are considered as holding a valid share. Applying for example the error-correcting codes decoding scheme described in Sect. 5.1 allows $\mathcal{C}$ to determine ω from the shares collected from the players of $H \cup D$.

Like Shamir's threshold scheme, Gennaro et al.'s VSS is perfect, i.e., the knowledge of $k-1$ or less shares (f_i, g_i) leaves ω completely undetermined.

6 Description of the Protocol

In this section we present our verifiable DOT protocol (See Fig. 1). The protocol is composed of five phases, described hereafter.

6.1 Phase 1 – Sharing of the Sender's Secrets

The sender $\mathcal{S}$ generates a quasi-random polynomial $P \in \mathbb{K}_{k-1}[X]$ such that

$$P = \omega_0 + \sum_{i=1}^{k-1} a_i X^i, \qquad \text{where } \ a_i \in_R \mathbb{K}, 1 \leq i \leq k-1.$$

Then, $\mathcal{S}$ generates the n-variate polynomial

$$Q(x, y_1, \ldots, y_{n-1}) = P(x) + \sum_{j=1}^{n-1} (\omega_j - \omega_0) y_j.$$

For each index $\ell \in \mathcal{I}_m$, $\mathcal{S}$ builds an $(n-1)$-variate polynomials $F_\ell = Q(\ell, y_1, \ldots, y_{n-1})$ and transmits this polynomial to the server S_ℓ. The sender does not intervene in the rest of the protocol.

At the end of this phase, m servers S_ℓ including up to $k-1$ malicious servers have received a polynomial F_ℓ.

6.2 Phase 2 – Sharing of the Receiver's Secret Inputs

The receiver $\mathcal{R}$ chooses the index $\sigma \in \{0, \ldots, n-1\}$ of the secret ω_σ she wishes to obtain as well as a set $\mathcal{I}_c \subset \mathcal{I}_m$ of $c \geq 4k-3$ indices of servers. The index σ is coded under the form of a vector $(\delta_\sigma^1, \ldots, \delta_\sigma^{n-1})$ of $n-1$ elements of $\{0, 1\}$.

Then, for each secret input δ_σ^s, $\mathcal{R}$ generates and distributes shares of δ_σ^s according to the VSS sharing phase of the protocol described in Sect. 5.2.

Let $S_1, \ldots, S_m$ be m servers.

Input The sender $\mathcal{S}$, contributes with n secrets $\omega_0, \ldots, \omega_{n-1} \in \mathbb{K}$.
The receiver $\mathcal{R}$, chooses an index $\sigma \in \{0, \ldots, n-1\}$, and contributes with $n-1$ private values $\delta_\sigma^1, \ldots, \delta_\sigma^{n-1} \in \{0, 1\}$

Output If $\mathcal{R}$ is detected as a cheater, then the protocol stops. Otherwise, if she follows the protocol, $\mathcal{R}$ receives ω_σ, while $\mathcal{S}$ receives nothing.

Phase 1 – **Sharing of the Sender's Secrets**

1. $\mathcal{S}$ generates an n-variate polynomial $Q(x, y_1, \ldots, y_{n-1}) = P(x) + \sum_{i=1}^{n-1} (\omega_i - \omega_0) y_i$ where $P \in \mathbb{K}_{k-1}[X]$ is a quasi-random polynomial whose constant term is ω_0.
2. $\mathcal{S}$ transmits to the server S_ℓ ($\ell \in \mathcal{I}_m = \{1, \ldots, m\}$) the $(n-1)$-variate polynomial $F_\ell(y_1, \ldots, y_{n-1}) = Q(\ell, y_1, \ldots, y_{n-1})$.

Phase 2 – **Sharing of the Receiver's Secret Inputs**

1. $\mathcal{R}$ chooses $\sigma \in \{0, \ldots, n-1\}$ and $\mathcal{I}_c \subset \mathcal{I}_m$, a set of $c \geq 4k-3$ servers to contact.
2. $\mathcal{R}$ generates a vector $\mathbf{\Theta} = (G_1, \ldots, G_{n-1})$ of $n-1$ quasi-random bivariate polynomials $G_s \in \mathbb{K}_{k-1}[X, Y]$, where the constant term of G_s is δ_σ^s. The polynomial $G_s(x, i)$ ($i \in \mathcal{I}_c$) is denoted $f_{s,i}(x)$ and the polynomial $G_s(i, y)$ ($i \in \mathcal{I}_c$) is denoted $g_{s,i}(y)$.
3. $\mathcal{R}$ generates c vectors $\boldsymbol{V_i} = (f_{1,i}, \ldots, f_{n-1,i})$ of polynomials ($i \in \mathcal{I}_c$). Similarly, $\mathcal{R}$ generates c vectors $\boldsymbol{W_i} = (g_{1,i}, \ldots, g_{n-1,i})$ of polynomials ($i \in \mathcal{I}_c$).
4. $\mathcal{R}$ transmits to S_i ($i \in \mathcal{I}_c$) the pair of vectors $(\boldsymbol{V_i}, \boldsymbol{W_i})$.
5. For $s = 1, \ldots, n-1$, S_i ($i \in \mathcal{I}_c$) sends to the server S_j ($j \in \mathcal{I}_c$) a random element $r_{s,i,j} \in_R \mathbb{K}$.

Phase 3 – **Detection of Cheaters**

1. S_i ($i \in \mathcal{I}_c$) broadcasts $f_{s,i}(j) + r_{s,i,j}$ and $g_{s,i}(j) + r_{s,j,i}$, for $s = 1, \ldots, n-1$. From the broadcast values and for $s = 1, \ldots, n-1$, S_i builds three sets of servers H_s, D_s and C_s, following the technique described in Sect. 5.2. If $|C_s| > k-1$, then $\mathcal{R}$ has cheated and the protocol stops.
2. S_i ($i \in \mathcal{I}_c$) determines $\mathcal{H} = \bigcap_{s=1}^{n-1} (H_s \cup D_s)$. If $c - |\mathcal{H}| \geq k$, then $\mathcal{R}$ has cheated and the protocol stops. Otherwise, the set of indices corresponding to the servers in $\mathcal{H}$ is denoted $\mathcal{I}_\mathcal{H}$.
3. S_i ($i \in \mathcal{I}_\mathcal{H}$) calculates $\boldsymbol{\Phi_i} = (Z_1(i), \ldots, Z_{n-1}(i))$. The share $Z_s(i)$ ($1 \leq s \leq n-1$) is calculated from the values $g_s(i)$ ($i \in \mathcal{I}_\mathcal{H}$) thanks to the error-correcting codes decoding scheme described in Sect. 5.1.

Phase 4 – **Computation of the Shares of the Chosen Secret**

Each server S_i ($i \in \mathcal{I}_\mathcal{H}$) calculates the share $\mu_i = F_i(\boldsymbol{\Phi_i})$ and sends it to $\mathcal{R}$.

Phase 5 – **Reconstruction of the Chosen Secret**

$\mathcal{R}$ interpolates a polynomial R of degree at most $k-1$, thanks to the error-correcting codes decoding scheme described in Sect. 5.1 and calculates $\omega_\sigma = R(0)$.

Figure 1: A verifiable $(4k-3,\ m)$-DOT-$\binom{n}{1}$ protocol

That is, $\mathcal{R}$ generates for $s = 1, \ldots, n-1$, a quasi-random bivariate polynomial $G_s \in \mathbb{K}_{k-1}[X,Y]$ such that $G_s(0,0) = \delta_\sigma^s$. For $i \in \mathcal{I}_c$, we denote $f_{s,i}(x)$ the polynomial $G_s(x,i)$ and $g_{s,i}(y)$ the polynomial $G_s(i,y)$. We also define, for $s = 1, \ldots, n-1$, the polynomial Z_s by $Z_s(y) = G_s(0,y)$. The polynomial $Z_s \in \mathbb{K}_{k-1}[X]$ satisfies the c equalities $Z_s(i) = f_{s,i}(0)$ $(i \in \mathcal{I}_c)$. The receiver $\mathcal{R}$ distributes to server S_i $(i \in \mathcal{I}_c)$ the two vectors of polynomials $\boldsymbol{V_i} = (f_{1,i}, \ldots, f_{n-1,i})$ and $\boldsymbol{W_i} = (g_{1,i}, \ldots, g_{n-1,i})$.

Then, for $s = 1, \ldots, n-1$, each contacted server S_i $(i \in \mathcal{I}_c)$ sends to the server S_j $(j \in \mathcal{I}_c)$ a random element $r_{s,i,j} \in_R \mathbb{K}$.

6.3 Phase 3 – Detection of Cheaters

This phase, composed of three steps, allows the servers to verify that the receiver is honest, possibly to detect and identify malicious servers, and above all to calculate the receiver's secret inputs shares.

6.3.1 Categorization of Servers.

Each server S_i $(i \in \mathcal{I}_c)$ broadcasts $f_{s,i}(j) + r_{s,i,j}$ and $g_{s,i}(j) + r_{s,j,i}$, for $s = 1, \ldots, n-1$. From the broadcast values, each server builds for each value $s = 1, \ldots, n-1$, c lists $L_{s,1}, \ldots, L_{s,c}$ where $L_{s,i}$ contains S_j if $f_{s,i}(j) + r_{s,i,j} \neq g_{s,j}(i) + r_{s,i,j}$.

Then, S_i transforms $L_{s,1}, \ldots, L_{s,c}$ into a graph $\mathcal{G}_s = (V_s, E_s)$ where $V_s = \{S_{i_1}, \ldots, S_{i_c}\}$ $(i_1, \ldots, i_c \in \mathcal{I}_c)$ and $(S_i, S_j) \in E_s$ if $S_i \notin L_{s,j}$ and $S_j \notin L_{s,i}$. The next step consists for each server S_i $(i \in \mathcal{I}_c)$ in building for $s = 1, \ldots, n-1$ the sets H_s, D_s and C_s. The set H_s is a group of servers in which any server's broadcast information was consistent with the broadcast information of all other servers of the group, D_s is a group of servers whose broadcast information was consistent with the broadcast information of at least $2k-1$ servers in H_s, and C_s is a set of servers detected as cheaters. Servers in D_s and C_s may be malicious or may be honest but have received forged shares from $\mathcal{R}$.

Using the algorithm described in Appendix A, S_i $(i \in \mathcal{I}_c)$ determines, for each value s $(s = 1, \ldots, n-1)$ a maximal matching M_s of $\overline{\mathcal{G}_s}$ and builds the set of servers $\overline{H_s} = \bigcup\{S_i, S_j\}$ where $(S_i, S_j) \in M_s$ and the set $H_s = V_s \setminus \overline{H_s}$. Then, S_i builds another set of servers D_s, thanks to the following algorithm: for each $S_u \notin H_s$, S_i checks the number n_u of pairs (S_u, S_v) in $\mathcal{G}_s$. If $n_u \geq 2k-1$ (i.e. at least k honest players have consistent information with S_u), then S_u is added to D_s. Otherwise, S_u is disqualified and added to the set C_s.

If S_i finds out that $|C_s| > k-1$, then the receiver $\mathcal{R}$ is considered as a cheater and the protocol stops.

6.3.2 Overall Verification.

Then, each server S_i $(i \in \mathcal{I}_c)$ agglomerates the sets constructed in the previous step. Thus, $\mathcal{H} = \bigcap_{s=1}^{n-1} (H_s \cup D_s)$. If S_i finds out that $c - |\mathcal{H}| > k-1$, then like in the previous step, the receiver $\mathcal{R}$ is considered as a cheater and the protocol stops.

The set of indices corresponding to the servers in $\mathcal{H}$ is denoted $\mathcal{I}_\mathcal{H}$. A server S_d belongs to the set $\mathcal{D}$ of disqualified servers if $d \in \mathcal{I}_c \setminus \mathcal{I}_\mathcal{H}$. The servers in $\mathcal{D}$ do not participate to the rest of the protocol.

6.3.3 Recovering of Shares.

Each server S_i $(i \in \mathcal{I}_\mathcal{H})$ calculates the vector

$$\boldsymbol{\Phi_i} = (Z_1(i), \ldots, Z_{n-1}(i)) \, .$$

The share $Z_s(i)$ $(1 \le s \le n-1)$ is calculated from the values $g_{s,j}(i)$ $(j \in \mathcal{I}_\mathcal{H})$ thanks to the error-correcting codes decoding scheme described in Sect. 5.1.

6.4 Phase 4 – Computation of the Shares of the Chosen Secret

Now, each server S_i $(i \in \mathcal{I}_\mathcal{H})$ holds an $(n-1)$-variate polynomial F_i as well as a vector $\boldsymbol{\Phi_i} = (Z_1(i), \ldots, Z_{n-1}(i))$ of secret inputs shares. Therefore, S_i is able to calculate the value $\mu_i = F_i(\boldsymbol{\Phi_i})$.

Each of the servers $S_i (i \in \mathcal{I}_\mathcal{H})$ transmits μ_i to $\mathcal{R}$.

6.5 Phase 5 – Reconstruction of the Chosen Secret

From the collected shares μ_i $(i \in \mathcal{H})$, the receiver $\mathcal{R}$ executes the error-correcting codes decoding scheme described in Sect. 5.1 to interpolate a polynomial R of degree at most $k-1$. Assuming $\mathcal{R}$ distributed shares of $\delta_\sigma^1, \ldots, \delta_\sigma^{n-1}$ in Phase 2, $\mathcal{R}$ can easily calculate $\omega_\sigma = R(0)$.

7 Security of the Protocol

In this section we show (proof sketch) that the proposed protocol satisfies all desirable conditions C_1', C_2 and C_3.

7.1 Correctness

Because we assume that the sender $\mathcal{S}$ is honest, each server S_i $(i \in \mathcal{I}_m)$ has received a correct polynomial F_i at the end of Phase 1.

Each of the $n-1$ sets $H_s \cup D_s$ $(1 \le s \le n-1)$ determined in the first step of Phase 3 contains at least $c - (k-1)$ servers holding k-consistent shares. If we set $t = k-1$, the number of servers in $H_s \cup D_s$ is at least $k + 2t$, including up to t malicious servers, because $c \ge 4k-3$. It follows that each server S_i of $H_s \cup D_s$, from the shares $g_{s,j}(i)$ (j such that $S_j \in H_s \cup D_s$), is able to interpolate a polynomial $f'_{s,i}$ thanks to the error-correcting codes decoding scheme described in Sect. 5.1 and to calculate the share $Z_s(i) = f'_{s,i}(0)$. If a server S_i was given an incorrect sharing polynomial $f_{s,i}$ by $\mathcal{R}$, then $f_{s,i}(0) \neq f'_{s,i}(0)$, but if a server S_i (honest or malicious) was given a correct share by $\mathcal{R}$, then $f_{s,i}(0) = f'_{s,i}(0)$.

However, the receiver may have cheated by giving inconsistent shares for one secret input to a group of servers and inconsistent shares for another secret input to another group of servers. This is why in the second step of Phase 3, each server S_i determines the set $C = \bigcup_{s=1}^{n-1} C_s$ composed of the cheaters regarding all the secret inputs. If $|C| > k-1$, then at least one honest server had incorrect shares, which means that this server received incorrect shares from $\mathcal{R}$. If $|C| \le k-1$, it cannot be proven that $\mathcal{R}$ cheated. It follows that $|\mathcal{H}| = c - |C| \ge c - (k-1)$. Again, we set $t = k-1$. Because $c \ge 4k-3$, it holds $|\mathcal{H}| \ge k + 2t$.

Therefore, at the end of Phase 3, each server $S_i \in \mathcal{H}$, i.e. at least $k + 2t$ servers, including up to t malicious servers, holds a vector $\boldsymbol{\Phi_i} = (Z_1(i), \ldots, Z_{n-1}(i))$ of k-consistent shares.

The computations executed by the servers $S_i \in \mathcal{H}$ in Phase 4 are therefore correct, although up to t of those servers may be malicious.

In Phase 5, $\mathcal{R}$ receives the responses from the servers of $\mathcal{H}$, i.e. at least $k+2t$ shares, including up to t incorrect shares. The receiver is then entitled to apply the error-correcting codes decoding scheme described in Sect. 5.1 and calculate the chosen secret $\omega_\sigma = R(0) = Q(0, \delta_\sigma^1, \ldots, \delta_\sigma^{n-1})$.

We conclude that condition C_1' is satisfied.

7.2 Receiver's Privacy

We show that the sender, but also the sender and $k-1$ corrupt servers, cannot obtain any information on the receiver's choice, along or after the protocol has been executed.

The index σ chosen by the receiver is represented under the form of a vector $(\delta_\sigma^1, \ldots, \delta_\sigma^{n-1})$ of private values. The receiver's input to the protocol consists of shares of these values. That is, each element δ_σ^s, which is either zero or one, is distributed among the c selected servers S_i $(i \in \mathcal{I}_c)$, using Gennaro et al.'s VSS scheme. This is achieved by generating a vector $(G_1, \ldots, G_{n-1})$ of $n-1$ quasi-random bivariate polynomials of $\mathbb{K}_{k-1}[X, Y]$, such that $G_s(0,0) = \delta_\sigma^s$.

Note that the polynomial $Z_s \in \mathbb{K}_{k-1}[Y]$ $(1 \le s \le n-1)$ defined by $Z_s = G_s(0, Y)$ is a quasi-random polynomial with a free coefficient $Z_s(0) = \delta_\sigma^s$. Thus, Z_s may be considered as a sharing polynomial for the value δ_σ^s.

For each private value δ_σ^s, a server S_i $(i \in \mathcal{I}_c)$ receives two polynomials $f_{s,i} = G_s(x, i)$ and $g_{s,i} = G_s(i, y)$. The server S_i is then able to calculate the share $Z_s(i)$ since $Z_s(i) = f_{s,i}(0)$. From the received polynomial $g_{s,i}$, S_i is also able to determine one share corresponding to the sharing polynomial $f_{s,j}$ received by the server S_j $(j \in \mathcal{I}_c)$. Indeed, S_i is able to calculate $g_{s,i}(j) = f_{s,j}(i)$.

In order to breach the privacy of the receiver along the execution of the protocol, or after completion of the protocol, a set of $k-1$ colluding servers S_ℓ $(\ell \in \mathcal{I}_b \subset \mathcal{I}_c)$ should be able to determine at least one of the values δ_σ^s. The set of $k-1$ collaborating servers, however, holds for each value δ_σ^s, $k-1$ shares associated with a Gennaro et al.'s scheme. More precisely, the coalition holds $k-1$ shares $Z_s(\ell)$ $(\ell \in \mathcal{I}_b)$ and also $k-1$ shares of each sharing polynomial $f_{s,j}$ of $\mathbb{K}_{k-1}[X]$ $(j \in \mathcal{I}_c)$. The knowledge of a k^{th} share of $f_{s,j}$ would allow to determine $Z_s(j)$ and by interpolation Z_s, and then $\delta_\sigma^s = Z_s(0)$. But, due to the perfectness of Gennaro et al.'s scheme, every set of $k-1$ shares provides the coalition with absolutely no information about the relevant private value.

Consequently, the condition C_2 is guaranteed.

7.3 Sender's Security

First, we show that along or after having executed the protocol, the receiver cannot obtain more than one secret and second, we demonstrate that a coalition of $k-1$ servers with the receiver is unable to obtain information on the secrets $\omega_0, \ldots, \omega_{n-1}$.

7.3.1 Sender's Security with Respect to the Receiver.

In the original protocol, if we denote $\mathcal{I}_k = \{x_1, \ldots, x_k\}$ the set of indices of the contacted servers, each collected share μ_i $(i \in \mathcal{I}_k)$ is actually the right member of the equation:

$$\omega_0 + \sum_{j=1}^{k-1} a_j i^j + \sum_{j=1}^{n-1} (\omega_j - \omega_0) Z_j(i) = \mu_i \ .$$

So, the receiver $\mathcal{R}$ is able to build the linear system of k equations in $n+k-1$ unknowns (n secrets, $k-1$ coefficients a_i):

$$\left\{\begin{array}{l}\sum_{j=1}^{k-1} a_j x_1^j + \omega_0(1-\sum_{j=1}^{n-1} Z_j(x_1)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_1) = \mu_{x_1} \\ \dotfill \\ \sum_{j=1}^{k-1} a_j x_k^j + \omega_0(1-\sum_{j=1}^{n-1} Z_j(x_k)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_k) = \mu_{x_k}\end{array}\right. \tag{1}$$

With our protocol, $\mathcal{R}$ collects additional shares. The $c \geq 4k-3$ collected shares are consistent, i.e., assuming $\mathcal{I}_c = \mathcal{I}_k \cup \{x_{k+1}, \ldots, x_c\}$, $\mathcal{R}$ is able to built the linear system

$$\left\{\begin{array}{l}\sum_{j=1}^{k-1} a_j x_1^j + \omega_0(1-\sum_{j=1}^{n-1} Z_j(x_1)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_1) = \mu_{x_1} \\ \dotfill \\ \sum_{j=1}^{k-1} a_j x_k^j + \omega_0(1-\sum_{j=1}^{n-1} Z_j(x_k)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_k) = \mu_{x_k} \\ \sum_{j=1}^{k-1} a_j x_{k+1}^j + \omega_0(1-\sum_{j=1}^{n-1} Z_j(x_{k+1})) + \sum_{j=1}^{n-1} \omega_j Z_j(x_{k+1}) = \mu_{x_{k+1}} \\ \dotfill \\ \sum_{j=1}^{k-1} a_j x_c^j + \omega_0(1-\sum_{j=1}^{n-1} Z_j(x_c)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_c) = \mu_{x_c}\end{array}\right. \tag{2}$$

According to Theorem 1 (See Appendix B), the two systems (1) and (2) are identical. It follows that $\mathcal{R}$ cannot learn more than one linear combination of secrets by collecting extra consistent shares. To force $\mathcal{R}$ to obtain information on no more than one secret from the linear combination, the technique described by Naor and Pinkas [14] may be applied; in Phase 1, $\mathcal{S}$ randomly selects n masks $c_0, \ldots, c_{n-1} \in \mathbb{K}^*$. Two polynomials P_1 and P_2 are generated: P_1 to share the n masked secrets $c_i\omega_i$ and P_2 to share the n masks c_i $(0 \leq i \leq n-1)$. Phase 3 is executed with the shares of both the masked secrets and the masks. In Phase 4, in response to the receiver's request $\boldsymbol{\Phi_i}$, each server S_i $(i \in \mathcal{I_H})$ returns two shares: $P_1(i, \boldsymbol{\Phi_i})$ and $P_2(i, \boldsymbol{\Phi_i})$. From the collected shares, $\mathcal{R}$ is able to apply the error-correcting codes decoding scheme described in Sect. 5.1 and calculate the chosen masked secret and its corresponding mask.

Consequently, like in the original protocol, $\mathcal{R}$ cannot obtain more than one secret.

7.3.2 Sender's Security with Respect to a Coalition of Servers.

Assuming the technique preventing $\mathcal{R}$ from obtaining information on more than one secret is used (See previous section), the initialisation phases of Blundo et al.'s and our protocols are the same. So at the end of Phase 1, a coalition of $k-1$ servers holds $2 \times (k-1)$ sharing $(n-1)$-variate polynomials. There is no advantage for the coalition to collude with the receiver to breach the sender's security, because the receiver has no input to contribute to an attack. It follows that the sender's security with respect to a coalition of the receiver and $k-1$ servers is the same for our protocol as for the original protocol, i.e. C_3.

References

[1] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC'88: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 1–10. ACM (1988)

[2] Berlekamp, E.: Algebraic coding theory revised 1984 edition. Aegean Park Press (1984)

[3] Blundo, C., D'Arco, P., De Santis, A., Stinson, D.R.: New results on unconditionally secure distributed oblivious transfer. In: Nyberg, K., Heys, H.M. (eds.) Selected Areas in Cryptography - SAC 2002. Lecture Notes in Computer Science, vol. 2595, pp. 291–309. Springer-Verlag (2003)

[4] Blundo, C., D'Arco, P., De Santis, A., Stinson, D.R.: On unconditionally secure distributed oblivious transfer. Journal of Cryptology 20(3), 323–373 (2007)

[5] Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC'88: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 11–19. ACM (1988)

[6] Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: SFCS'85: Proceedings of the 26th Annual Symposium on Foundations of Computer Science. pp. 383–395. IEEE Computer Society (1985)

[7] Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT'99. Lecture Notes in Computer Science, vol. 1592, pp. 311–326. Springer-Verlag (1999)

[8] Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 316–334. Springer-Verlag (2000)

[9] D'Arco, P., Stinson, D.R.: On unconditionally secure robust distributed key distribution centers. In: Zheng, Y. (ed.) Advances in Cryptology - ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 181–189. Springer-Verlag (2002)

[10] Desmedt, Y., Kurosawa, K., Van Le, T.: Error correcting and complexity aspects of linear secret sharing schemes. In: Boyd, C., Mao, W. (eds.) Information Security - ISC 2003. Lecture Notes in Computer Science, vol. 2851, pp. 396–407. Springer-Verlag (2003)

[11] Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics 17(3), 449–467 (1965)

[12] Gao, S.: A new algorithm for decoding Reed-Solomon codes. In: Bhargava, V.K., Poor, H.V., Tarokh, V., Yoon, S. (eds.) Communications, Information and Network Security, pp. 55–68. Kluwer Academic Publishers (2003)

[13] Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: STOC'01: Proceedings of the thirty-third annual ACM symposium on Theory of computing. pp. 580–589. ACM (2001)

[14] Naor, M., Pinkas, B.: Distributed oblivious transfer. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000. Lecture Notes in Computer Science, vol. 1976, pp. 205–219. Springer-Verlag (2000)

[15] Nikov, V., Nikova, S., Preneel, B., Vandewalle, J.: On unconditionally secure distributed oblivious transfer. In: Menezes, A., Sarkar, P. (eds.) Advances in Cryptology - INDOCRYPT 2002. Lecture Notes in Computer Science, vol. 2551, pp. 395–408. Springer-Verlag (2002)

[16] Rabin, T.: Robust sharing of secrets when the dealer is honest or cheating. J. ACM 41(6), 1089–1109 (1994)

[17] Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Johnson, D.S. (ed.) STOC'89: Proceedings of the twenty-first annual ACM symposium on Theory of computing. pp. 73–85. ACM (1989)

[18] Reed, I., Solomon, G.: Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics 8(2), 300–304 (1960)

[19] Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)

[20] Stinson, D.R., Wei, R.: Unconditionally secure proactive secret sharing scheme with combinatorial structures. In: Heys, H., Adams, C. (eds.) Selected Areas in Cryptology - SAC'99. Lecture Notes in Computer Science, vol. 1758, pp. 200–214. Springer-Verlag (2000)

[21] Tompa, M., Woll, H.: How to share a secret with cheaters. Journal of Cryptology 1(2), 133–138 (1988)

[22] Zhong, S., Richard Yang, Y.: Verifiable distributed oblivious transfer and mobile agent security. In: Proceedings of the 2003 joint workshop on Foundations of mobile computing. pp. 12–21. ACM (2003)

[23] Zhong, S., Richard Yang, Y.: Verifiable distributed oblivious transfer and mobile agent security. Mobile Networks and Applications 11(2), 201–210 (2006)

Appendix

A Subprotocol Maximal Matching in a Graph

A *graph* $\mathcal{G}$ is defined as a pair (V, E) where $V = \{v_1, \ldots, v_n\}$ ($n \in \mathbb{N}$) is a set of vertices or nodes and $E = \{e_1, \ldots, e_m\}$ ($m \in \mathbb{N}$), is a set of edges, where an edge is a pair of vertices (v_i, v_j) ($v_i \in V$, $v_j \in V$, $v_i \neq v_j$).

The graph $\bar{\mathcal{G}} = (\bar{V}, \bar{E})$ is the *complement* of $\mathcal{G}$ if $\bar{V} = V$ and $e \in \bar{E}$ if and only if $e \notin E$.

In a graph $\mathcal{G} = (V, E)$, a *matching* $M \subset E$ is a set of edges such that no two edges of M have a vertex in common.

A *maximal matching* is a matching which cannot be extended by the addition of any edge of $\mathcal{G}$.

To determine a maximal matching M of a graph $\mathcal{G} = (V, E)$, a simple greedy algorithm may be used: First, an edge is selected, added to M and its vertices are deleted from $\mathcal{G}$. In the remaining subgraph, another edge is selected, added to M and its vertices are deleted from the subgraph. This process continues until no edge is available.

B Additional Consistent Shares in Blundo et al.'s Protocol

To demonstrate that a receiver $\mathcal{R}$ who collects $c \geq k$ consistent shares does not obtain more linear combinations of secrets than if she collects k shares, it is sufficient to demonstrate Theorem 1.

Theorem 1. *The linear systems* (1) *and* (2) *determined in Sect. 7.3.1 are identical.*

Proof. System 2 can be written under the matrix form $\boldsymbol{\mathcal{A}x} = \boldsymbol{b}$, where

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} x_1 & \ldots & x_1^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_1)) & Z_1(x_1) & \ldots & Z_{n-1}(x_1) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_k & \ldots & x_k^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_k)) & Z_1(x_k) & \ldots & Z_{n-1}(x_k) \\ x_{k+1} & \ldots & x_{k+1}^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_{k+1})) & Z_1(x_{k+1}) & \ldots & Z_{n-1}(x_{k+1}) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_c & \ldots & x_c^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_c)) & Z_1(x_c) & \ldots & Z_{n-1}(x_c) \end{pmatrix},$$

$$\boldsymbol{x} = \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \\ \omega_0 \\ \vdots \\ \omega_{n-1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} \mu_{x_1} \\ \vdots \\ \mu_{x_k} \\ \mu_{x_{k+1}} \\ \vdots \\ \mu_{x_c} \end{pmatrix}$$

We show that each row ℓ ($k + 1 \leq \ell \leq c$) of $\boldsymbol{\mathcal{A}}$ is a linear combination of the rows 1 to k.

Let us define the $k + n - 1$ polynomials $T_i \in \mathbb{K}_{k-1}[X]$ ($1 \leq i \leq k + n - 1$) by the relations:

- $T_i = X^i$ for $i = 1, \ldots, k - 1$,
- $T_k = 1 - \sum_{j=1}^{n-1} Z_j$,
- $T_{k+s} = Z_s$ for $s = 1, \ldots, n - 1$.

The matrix $\boldsymbol{\mathcal{A}}$ may be written under the form:

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} T_1(x_1) & \dots & T_{k-1}(x_1) & T_k(x_1) & T_{k+1}(x_1) & \dots & T_{k+n-1}(x_1) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ T_1(x_c) & \dots & T_{k-1}(x_c) & T_k(x_c) & T_{k+1}(x_c) & \dots & T_{k+n-1}(x_c) \end{pmatrix},$$

In $\mathbb{K}_{k-1}[X]$, we consider the k Lagrange polynomials L_{x_i} related to $x_1, \dots, x_k$ (these elements of $\mathbb{K}$ are distinct and different from 0):

$$L_{x_i} = \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{X - x_j}{x_i - x_j}$$

Because $(L_{x_1}, \dots, L_{x_k})$ is a basis of the vector space of polynomials of degree at most $k-1$, each polynomial $T_i \in \mathbb{K}_{k-1}[X]$ can be written as $T_i = \sum_{j=1}^{k} T_i(x_j) L_{x_j}$. It follows that the matrix $\boldsymbol{\mathcal{A}}$ is:

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} T_1(x_1) & \dots & T_{k+n-1}(x_1) \\ \vdots & \dots & \vdots \\ T_1(x_k) & \dots & T_{k+n-1}(x_k) \\ \sum_{j=1}^{k} T_1(x_j) L_{x_j}(x_{k+1}) & \dots & \sum_{j=1}^{k} T_{k+n-1}(x_j) L_{x_j}(x_{k+1}) \\ \vdots & & \vdots \\ \sum_{j=1}^{k} T_1(x_j) L_{x_j}(x_c) & \dots & \sum_{j=1}^{k} T_{k+n-1}(x_j) L_{x_j}(x_c) \end{pmatrix}.$$

Clearly, the lines $k+1, ..., c$ are linear combinations of the first k lines.

It follows that the matrix equality $\boldsymbol{\mathcal{A}x} = \boldsymbol{b}$ can be written $\boldsymbol{\mathcal{A}'x} = \boldsymbol{b'}$, where

$$\boldsymbol{\mathcal{A}'} = \begin{pmatrix} x_1 & \dots & x_1^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_1)) & Z_1(x_1) & \dots & Z_{n-1}(x_1) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_k & \dots & x_k^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_k)) & Z_1(x_k) & \dots & Z_{n-1}(x_k) \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{b'} = \begin{pmatrix} \mu_{x_1} \\ \vdots \\ \mu_{x_k} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The linear system corresponding to this equation is exactly (1), which concludes the demonstration.

□