Security Analysis of Polynomial Interpolation-Based Distributed Oblivious Transfer Protocols

Christian L.F. Corniaux(✉) and Hossein Ghodosi

James Cook University, Townsville 4811, Australia
chris.corniaux@my.jcu.edu.au, hossein.ghodosi@jcu.edu.au

Abstract. In an unconditionally secure *Distributed Oblivious Transfer* (DOT) protocol, a receiver contacts at least k servers to obtain one of the n secrets held by a sender. Once the protocol has been executed, the sender does not know which secret was chosen by the receiver and the receiver has not gained information on the secrets she did not choose. In practical applications, the probability distribution of the secrets may not be uniform, e.g., when DOT protocols are used in auctions, some bids may be more probable than others.

In this kind of scenario, we show that the claim "a party cannot obtain more than a linear combination of secrets" is incorrect; depending on the probability distribution of the secrets, some existing polynomial interpolation-based DOT protocols allow a cheating receiver, or a curious server, who has obtained a linear combination of the secrets to determine all the secrets.

Keywords: Cryptographic protocol · Distributed Oblivious Transfer · Linear combination of secrets · Probability distribution · Unconditional security

1 Introduction

Unconditionally secure *Distributed Oblivious Transfer* (DOT) protocols allow a receiver to obtain one of the n secrets held by a sender (see for example [3,8,9]), like *Oblivious Transfer* (OT) protocols. But, unlike in OT protocols, the sender and the receiver do not directly interact with each other; the sender distributes information on his secrets to m servers and the receiver contacts k of them to collect enough data to determine the secret she wishes to obtain.

The security level of a DOT protocol is characterized by the threshold parameter k, corresponding to the minimum number of servers the receiver has to interact with, to obtain the chosen secret. The protocol itself is composed of two phases. In a first phase (the *set-up phase*), the sender distributes parts — called *shares* — of the secrets to the servers and does not intervene in the rest of the protocol. In a second phase (the *transfer phase*), the receiver selects the index of a secret, sends shares of this index to t servers ($k \leq t \leq m$) and receives back t

J. Lee and J. Kim (Eds.): ICISC 2014, LNCS 8949, pp. 363–380, 2015.
DOI: 10.1007/978-3-319-15943-0_22

shares allowing her to reconstruct the chosen secret. The security of a DOT protocol may be assessed thanks to the following informal security conditions based on definitions given by Blundo, D'Arco, De Santis and Stinson [2,3]:

C_1. Correctness – The receiver is able to determine the chosen secret once she has received information from t contacted servers ($t \geq k$).

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

C_3. Sender's privacy with respect to λ_S servers and the receiver – A coalition of up to λ_S servers ($1 \leq \lambda_S \leq k-1$) 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 – Once the protocol has been executed, a coalition of up to λ_C dishonest servers ($0 \leq \lambda_C \leq k-1$) and the receiver does not obtain any information about secrets which were not chosen by the receiver. This security condition may be decomposed into two parts; Given the transcript of the interaction with t servers ($t \geq k$),

$C_{4.1}$. The receiver does not obtain any information about secrets she did not choose ($\lambda_C = 0$).

$C_{4.2}$. A coalition of up to λ_C dishonest servers and the receiver does not obtain any information about secrets which were not chosen by the receiver ($\lambda_C > 0$).

Blundo et al. [3] define a DOT protocol as *private* if the following security conditions are satisfied: C_1, for $t = k$, C_2, for $\lambda_R = k-1$, C_3, for $\lambda_S = k-1$ and $C_{4.1}$. A DOT protocol is defined as *strongly private* if it is private and if condition $C_{4.2}$ is satisfied for $\lambda_C = k-1$. Blundo et al. have shown that one-round polynomial interpolation-based DOT protocols cannot reach strong privacy, i.e., if $t = k$ and $\lambda_R = k-1$, then condition $C_{4.2}$ cannot be satisfied for $\lambda_C = k-1$ (a round is a set of consistent requests/responses exchanged between the receiver and t servers in the transfer phase). More generally, Nikov, Nikova, Preneel and Vandewalle [9] have demonstrated that the relation $\lambda_R + \lambda_C < k$ needs to be satisfied for conditions C_2 and $C_{4.2}$ to be guaranteed.

In their OT protocol allowing a receiver to obtain one of the n secrets held by a sender, Brassard, Crépeau and Robert [4] note that it should be impossible for the receiver to gain joint information on the secrets held by the sender. This remark is the consequence that, in classical cryptography, shifting the letters of an English message thanks to a secret word is not secure. For example, an adversary can break the Vigenère cryptosystem, which basically produces a cryptogram by shifting the letters of a message according to a secret key, in plain English too. The non-uniform repartition of letters in the original message, and in the key for some variants of Vigenère's cryptosystem, allows an adversary to retrieve both the original text and the key from a cryptogram (*index of coincidence* technique [6] and *Kasiski method* [7]). Aware of this weakness, OT and DOT protocols' designers have taken a great care to construct protocols where receivers cannot learn a linear combination of the secrets held by a sender.

In some instances, where the secrets are elements randomly selected in a finite field, the precaution is useless. But when the probability distribution of

each secret is not uniform (like for letters in English messages), the control is essential.

In some polynomial interpolation-based DOT protocols, e.g. [3,8], it is claimed that a party "cannot learn more than a linear combination of secrets". In this paper, we show that such claims are incorrect. Overall, the knowledge of a linear combination of secrets combined with the knowledge of the probability distributions of the same secrets may lead to the knowledge of the secrets themselves. In other words, if a curious server obtains a linear combination of secrets, security condition C_3 cannot be satisfied and similarly, if a curious or malicious receiver obtains a linear combination of secrets, security condition $C_{4.1}$ cannot be satisfied. In addition, we demonstrate that in Blundo et al.'s sparse polynomial interpolation-based DOT protocol, the two techniques preventing the servers and the receiver from learning linear combinations of secrets make the protocol insecure (in spite of Blundo et al.'s claim, security condition $C_{4.1}$ is not satisfied); indeed, only one of the two techniques should be applied to guarantee the security of the protocol.

The organization of the paper is as follows. In Sect. 2 we introduce a few notations and show that the combined knowledge of the probability distributions of secrets and of a linear combination of these secrets may lead to the knowledge of all of them. Then, in Sect. 3, we shortly describe the general form of polynomial interpolation-based DOT protocols and show how in these protocols a receiver is able to obtain a linear combination of secrets. Section 4 is devoted to the analysis of some protocols [1,2,8,9] in the light of the previous section. In Sect. 5, we show that even Blundo et al.'s sparse polynomial interpolation-based DOT protocol [3], designed to protect the sender's privacy against a malicious receiver in presence of honest servers, does not satisfy security condition $C_{4.1}$. Our conclusion follows in Sect. 6.

2 Preliminaries

2.1 Notations and Definitions

The settings of the different DOT protocols described in this paper encompass a sender $\mathcal{S}$ who owns n secrets $\omega_1, \omega_2, \ldots, \omega_n$ ($n > 1$), a receiver $\mathcal{R}$ who wishes to learn a secret ω_σ, and m servers S_j ($j \in \mathcal{I}_m$ where $\mathcal{I}_m \subset \mathbb{N}$ is a set of $m \geq 2$ indices).

The protocols require the availability of private communication channels between the sender and the servers and between the receiver and the servers. 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.

All operations are executed in a finite field $\mathbb{K} = \mathbb{F}_p$ (p prime, $p > 2$). We assume that $p > \max(n, \omega_1, \omega_2, \ldots, \omega_n, m)$. By an abuse of language, a polynomial and its corresponding polynomial function will not be differentiated. We denote $\mathbb{K}^*$ the set $\mathbb{K} \setminus \{0\}$, $[n]$ the set of natural numbers (or elements of the prime finite field $\mathbb{K}$) $\{1, 2, \ldots, n\}$, and δ_i^j the Kronecker's symbol, equal to 0 if

$i \neq j$ and to 1 if $i = j$. If $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ are two n-tuples of elements of $\mathbb{K}$, we define $\boldsymbol{u} \bullet \boldsymbol{v} = \sum_{i=1}^{n} u_i \times v_i$.

We also formally define a *quasi-random* polynomial.

Definition 1. *If* $(\mathbb{K}[X], +, \times)$ *is the ring of polynomials over* $\mathbb{K}$ *and* $(\mathbb{K}_d[X], +)$ *the additive group of polynomials of degree at most* d *over* $\mathbb{K}$*, we say that a polynomial* $F = \sum_{i=0}^{d} f_i X^i$ *of* $\mathbb{K}_d[X]$ *is* quasi-random*, if the coefficients* f_i $(1 \leq i \leq d)$ *are randomly selected in* $\mathbb{K}$ *and the constant term* $f_0 \in \mathbb{K}$ *has a predefined value.*

In addition, we denote p_ω the probability mass function associated with the secret ω taken in the finite field $\mathbb{K}$.

2.2 Linear Combination of Two Secrets

In some DOT protocols, e.g. [3,8], it is claimed that the receiver cannot learn more than a linear combination of secrets. Actually, the knowledge of a linear combination of secrets combined with the knowledge of the probability distributions of the same secrets may lead to the knowledge of the secrets themselves, as shown in the following basic example.

Example 1. Let ω_1 and ω_2 be two secrets in the prime finite field $\mathbb{K} = \mathbb{F}_{11}$ with the probability distributions:

$$\begin{cases} p_{\omega_1}(0) = 0.5, p_{\omega_1}(1) = 0.5, p_{\omega_1}(i) = 0 \text{ if } i \neq 0 \text{ and } i \neq 1, \\ p_{\omega_2}(0) = 0.5, p_{\omega_2}(3) = 0.5, p_{\omega_2}(i) = 0 \text{ if } i \neq 0 \text{ and } i \neq 3. \end{cases}$$

We assume that a party is able to determine, for instance, the linear combination $\ell = \omega_2 - \omega_1$. From the probability distributions of the secrets, the only possible values of ℓ are:

$$\begin{cases} 0, & \text{if } \omega_1 = 0 \text{ and } \omega_2 = 0, \\ 3, & \text{if } \omega_1 = 0 \text{ and } \omega_2 = 3, \\ 10, & \text{if } \omega_1 = 1 \text{ and } \omega_2 = 0, \text{ and} \\ 2, & \text{if } \omega_1 = 1 \text{ and } \omega_2 = 3. \end{cases}$$

In this scenario where all the potential values of ℓ are different, the party just has to compare ℓ with the values 0, 3, 10 and 2 to determine the secrets ω_1 and ω_2.

2.3 Linear Combination of Secrets

More generally, we have

Lemma 1. *Let* $\omega_1, \omega_2, \ldots, \omega_n$ *be* n *secrets in* $\mathbb{K} = \mathbb{F}_p$ *(*p *prime) and* s *be the integer such that* $2^s \leq p < 2^{s+1}$*. If* $n \leq s$ *and* $\ell \in \mathcal{V} = \{0, 1, \ldots, 2^n - 1\} \subset \mathbb{K}$*, there exists probability distributions* $p_{\omega_1}, p_{\omega_2}, \ldots, p_{\omega_n}$ *such that one and only one* n*-tuple of secrets satisfies the linear combination* $\ell = \omega_1 + \omega_2 + \ldots + \omega_n$*.*

Proof. Given an element ℓ of $\mathbb{K}$ such that $0 \leq \ell \leq 2^n - 1$, we just have to exhibit n probability distributions $p_{\omega_1}, p_{\omega_2}, \ldots, p_{\omega_n}$, such that only one n-tuple $(\omega_1, \omega_2, \ldots, \omega_n)$ allows the linear combination $\ell = \omega_1 + \omega_2 + \ldots + \omega_n$ to be satisfied.

We define the probability distribution of the secret ω_i $(1 \leq i \leq n)$ by

$$p_{\omega_i}(j) = \begin{cases} 0.5, \text{ if } j = 0 \\ 0.5, \text{ if } j = 2^{i-1} \\ 0, \quad \text{otherwise.} \end{cases}$$

If $\ell \in \mathcal{V}$, let $B^\ell = b^\ell_{n-1} b^\ell_{n-2} \ldots b^\ell_1 b^\ell_0$ be the unique binary representation of ℓ. The n-tuple $\left(b^\ell_{n-1}, b^\ell_{n-2}, \ldots, b^\ell_1, b^\ell_0\right)$ is denoted β^ℓ and the set $\mathcal{U}$ is defined by $\mathcal{U} = \{\beta^0, \beta^1, \ldots, \beta^{2^n-1}\}$. We also define the function

$$\begin{aligned} f \colon \mathcal{V} &\longrightarrow \mathcal{U} \\ \ell &\longmapsto \beta^\ell \end{aligned}$$

The sets $\mathcal{U}$ and $\mathcal{V}$ have the same size (2^n elements) and the function f is injective, since every element $\ell \in \mathcal{V}$ has a unique binary representation; therefore f is bijective. We conclude that for any element $\ell \in \mathcal{V}$, there exist a unique n-tuple $\left(b^\ell_{n-1}, b^\ell_{n-2}, \ldots, b^\ell_1, b^\ell_0\right)$ where $b^{(\ell)}_i \in \{0, 1\}$ for $i = 0, 1, \ldots, n-1$ such that ℓ is written as a linear combination of the secrets $\omega_1, \omega_2, \ldots, \omega_n$:

$$\begin{aligned} \ell &= b^\ell_{n-1} \times 2^{n-1} + b^\ell_{n-2} \times 2^{n-2} + \ldots + b^\ell_1 \times 2^1 + b^\ell_0 \times 2^0 \\ &= \omega_n + \omega_{n-1} + \ldots + \omega_2 + \omega_1 \end{aligned}$$

We conclude that $\omega_i = b^\ell_{i-1} \times 2^{i-1}$ for $i = 1, 2, \ldots, n$. □

Using this basic result, we review in the next section some polynomial interpolation-based DOT protocols and show that their security is weaker than expected.

3 Polynomial Interpolation-Based DOT Protocols

Each of the existing unconditional secure polynomial interpolation-based DOT protocols, for example [1–3,5,8,9], follows the same principle.

– Before the protocol is executed, some details are made public: the number n of secrets, the threshold parameter k, the sender's privacy parameter λ_S, the sender's strong privacy parameter λ_C, the receiver's privacy parameter λ_R, the meaning of each secret ω_i $(1 \leq i \leq n)$, the joint probability $p_{\omega_1, \omega_2, \ldots, \omega_n}$ of the secrets, the encoding parameter e $(e > 0)$, the encoding function E where $E \colon [n] \longrightarrow \mathbb{K}^e$ encodes the index chosen by the receiver, the hiding parameter N corresponding to the number of monomials of the hiding polynomial Q (see set-up phase below) before reduction and N e-variate polynomials V_i $(1 \leq i \leq N)$ of $\mathbb{K}[Y_1, Y_2, \ldots, Y_e]$. The degree of the multivariate polynomial V_i is v_i; it is the highest degree of the monomials of V_i, assuming that the degree of a monomial is the sum of the degrees of its variables.

– In the set-up phase, the sender $\mathcal{S}$ generates N quasi-random polynomials U_i $(1 \leq i \leq N)$ of $\mathbb{K}_{u_i}[X]$ $(u_i \leq k-1)$ such that $\sum_{i=1}^{N} U_i(0)V_i(E(j)) = \omega_j$, for $j = 1, 2, \ldots, n$. The free coefficient of U_i is $U_i(0) = a_{i,0} + \sum_{j=1}^{n} a_{i,j}\omega_j$ where the coefficients $a_{i,j}$ $(0 \leq j \leq n)$ are randomly selected in $\mathbb{K}$. Then, $\mathcal{S}$ builds an $(e+1)$-variate polynomial:

$$Q(x, y_1, y_2, \ldots, y_e) = \sum_{i=1}^{N} U_i(x) \times V_i(y_1, y_2, \ldots, y_e),$$

and distributes the N-tuple $\boldsymbol{u}_j = (\, U_1(j), U_2(j), \ldots, U_N(j)\,)$ to the server S_j $(j \in \mathcal{I}_m)$.

– In the oblivious transfer phase, the receiver $\mathcal{R}$ who wishes to obtain the secret ω_σ prepares an e-tuple $E(\sigma) = (\, q_1, q_2, \ldots, q_e\,)$ as well as e quasi-random polynomial Z_i $(1 \leq i \leq e)$ of $\mathbb{K}_{\lambda_R}[X]$ $(\lambda_R \leq k-1)$ such that $Z_i(0) = q_i$. Then, $\mathcal{R}$ selects a subset $\mathcal{I}_t \subset \mathcal{I}_m$ of t indices $(k \leq t \leq m)$ and sends to each server S_j $(j \in \mathcal{I}_t)$ the request $\boldsymbol{z}_j = (\, Z_1(j), Z_2(j), \ldots, Z_e(j)\,)$. On reception of $\boldsymbol{z}_j$, S_j calculates and returns $\boldsymbol{u}_j \bullet \boldsymbol{v}_j$ to $\mathcal{R}$, where $\boldsymbol{v}_j = (\, V_1(\boldsymbol{z}_j), V_2(\boldsymbol{z}_j), \ldots, V_N(\boldsymbol{z}_j)\,)$. Because the relation $u_i + \lambda_R \times v_i \leq k-1$ is satisfied for $i = 1, 2, \ldots, N$, the receiver $\mathcal{R}$ is able to interpolate a polynomial $R \in \mathbb{K}_{k-1}[X]$ from the t pairs $(\, i_j, \boldsymbol{u}_j \bullet \boldsymbol{v}_j\,)$ and to calculate $\omega_\sigma = R(0)$.

The characteristics of the sparse polynomial interpolation-based DOT protocols analysed hereafter ([1–3,8,9]) are described in Annex A.

We note that if the receiver $\mathcal{R}$ does not follow the protocol and prepares, instead of $E(\sigma)$, the e-tuple $(\, \gamma_1, \gamma_2, \ldots, \gamma_e\,)$ such that $V_i(\gamma_1, \gamma_2, \ldots \gamma_e) = \alpha_i$ $(1 \leq i \leq N)$, then she is able to compute

$$\begin{aligned} R(0) &= \sum_{i=1}^{N} U_i(0)V_i(\gamma_1, \gamma_2, \ldots, \gamma_e) \\ &= \sum_{i=1}^{N} \Big(\alpha_i\big(a_{i,0} + \sum_{j=1}^{n} a_{i,j}\omega_j\big)\Big) \\ &= \sum_{i=1}^{N} \alpha_i a_{i,0} + \sum_{j=1}^{n} \Big(\sum_{i=1}^{N} \alpha_i a_{i,j}\Big)\omega_j \end{aligned}$$

Consequently, if $\mathcal{R}$ is able to determine an e-tuple $(\, \gamma_1, \gamma_2, \ldots, \gamma_e\,)$ such that $\sum_{i=1}^{N} \alpha_i a_{i,0} = 0$, she obtains a linear combination of the secrets $\omega_1, \omega_2, \ldots, \omega_n$.

In addition, if the values $\alpha_1, \alpha_2, \ldots, \alpha_N$ resulting from the choice of $(\gamma_1, \gamma_2, \ldots, \gamma_e)$ are such that $\sum_{i=1}^{N} \alpha_i a_{i,j} = 1$, for $j = 1, 2, \ldots, n$, then $\mathcal{R}$ is able to establish the environment of the scenario of Sect. 2.3.

4 Weaknesses of Some DOT Protocols

4.1 Protocols Insecure Against Curious Servers

In 2000, Naor and Pinkas [8] introduced a sparse polynomial interpolation-based unconditionally secure DOT protocol where the sender $\mathcal{S}$ holds two secrets ω_1

and ω_2. In this protocol, the hiding parameter N described in Sect. 3 is $N = 2$ and the polynomials U_1 and U_2 are:

$$U_1(x) = \omega_1 + \sum_{i=1}^{k-1} a_{1,i} x^i$$

and

$$U_2(x) = \omega_2 - \omega_1$$

where the coefficients $a_{1,i}$ $(1 \leq i \leq k-1)$ are randomly selected in $\mathbb{K}$ (see Annex A.1). It follows that in the set-up phase, each server S_j $(j \in \mathcal{I}_m)$ receives a pair $\boldsymbol{u}_j = (U_1(j), U_2(j) = \omega_2 - \omega_1)$ from $\mathcal{S}$. That is, every server receives a linear combination of secrets and may (see Lemma 1) determine both secrets. Consequently, security condition C_3 is not guaranteed.

The sparse polynomial interpolation-based unconditionally secure DOT protocol proposed by Blundo et al. [2] in 2002 is an extension of Naor and Pinkas's protocol to n secrets $\omega_1, \omega_2, \ldots, \omega_n$ where $n \geq 2$. The hiding parameter is $N = n$, the free coefficient of U_1 is $U_1(0) = \omega_1$ and the free coefficient of U_i $(2 \leq i \leq n)$ is $U_i(0) = \omega_i - \omega_1$ (see Annex A.2). Each server S_j $(j \in \mathcal{I}_m)$ receives an n-tuple $\boldsymbol{u}_j = (U_1(j), \omega_2 - \omega_1, \omega_3 - \omega_1, \ldots, \omega_n - \omega_1)$ in the set-up phase and thus holds linear combinations of the secrets. Again, according to Lemma 1, each server may determine all the secrets of the sender and security condition C_3 is not satisfied.

Another polynomial interpolation-based unconditionally secure DOT protocol was introduced in 2002 by Nikov et al. [9]. In this protocol, like in Blundo et al.'s protocol, the hiding parameter is $N = n$, the free coefficient of U_1 is $U_1(0) = \omega_1$ and the free coefficient of U_i $(2 \leq i \leq n)$ is $U_i(0) = \omega_i - \omega_1$. The polynomial U_1 belongs to $\mathbb{K}_{k-1}[X]$ and the polynomials $U_2, U_3, \ldots, U_n$ belong to $\mathbb{K}_{\lambda_C}[X]$ (see Annex A.4). We note that if $\lambda_S = \lambda_C = 0$, which is not allowed with our security model since $\lambda_S \geq 1$ (see security parameters in Sect. 1), each server S_j $(j \in \mathcal{I}_m)$ receives an n-tuple $\boldsymbol{u}_j = (U_1(j), \omega_2 - \omega_1, \omega_3 - \omega_1, \ldots, \omega_n - \omega_1)$ in the set-up phase. Again, according to Lemma 1, each server may determine all the secrets of the sender.

Similarly, in the interpolation-based unconditionally secure DOT protocol constructed from a private information retrieval protocol presented by Beimel, Chee, Wang and Zhang [1], each server may determine all the secrets of the sender if $\lambda_S = \lambda_C = 0$. In this case, the hiding parameter is $N = n + 1$, the free coefficient of U_1 is $U_1(0) = a_{1,0}$, a random element of $\mathbb{K}$ and the free coefficient of U_i $(2 \leq i \leq n+1)$ is $U_i(0) = \omega_{i-1} - a_{1,0}$. The polynomial U_1 belongs to $\mathbb{K}_{k-1}[X]$ and the polynomials $U_2, U_3, \ldots, U_n$ belong to $\mathbb{K}_{\lambda_C}[X]$ (see Annex A.5). Again, in our security model, security condition C_3 is not guaranteed.

4.2 Protocols Insecure Against a Greedy Receiver

In the sparse polynomial interpolation-based unconditionally secure DOT protocol introduced by Naor and Pinkas [8], the encoding function is $E(\sigma) = \left(1, \delta_\sigma^1\right)$

and the polynomials V_1 and V_2 are $V_1(y_1, y_2) = 1$ and $V_2(y_1, y_2) = y_2$. As mentioned in the previous section, the free coefficient of U_1 is $U_1(0) = \omega_1$. If a cheating receiver sends a request[1] $\boldsymbol{z}_j = (1, 1/2)$ to each server S_j $(j \in \mathcal{I}_t)$, it receives back $\sum_{i=1}^{2} U_i(j)V_i(\boldsymbol{z}_j) = U_1(j) + 1/2(\omega_2 - \omega_1)$. Interpolating a polynomial R from $t \geq k$ collected values, the receiver calculates $R(0) = U_1(0) + 1/2(\omega_2 - \omega_1) = 1/2(\omega_2 + \omega_1)$. From this linear combination, the receiver may (see Lemma 1) determine both secrets. Consequently, security condition $C_{4.1}$ is not guaranteed.

The insecurity is the same in Blundo et al.'s protocol [2]; the receiver sends the n-tuple request $\boldsymbol{z}_j = (1, 1/n, 1/n, \ldots, 1/n)$ to each server S_j $(j \in \mathcal{I}_t)$. The linear combination determined by the receiver is then $R(0) = 1/n \sum_{i=1}^{n} \omega_i$. Like in Naor and Pinkas's protocol, security condition $C_{4.1}$ is not guaranteed.

In the DOT protocol introduced by Nikov et al. [9], the free coefficient of U_1 is $U_1(0) = \omega_1$ and the free coefficient of U_i $(2 \leq i \leq n)$ is $U_i(0) = \omega_i - \omega_1$, exactly like in Blundo et al.'s protocol. Therefore, with the same n-tuple request $\boldsymbol{z}_j = (1, 1/n, 1/n, \ldots, 1/n)$ as above sent to servers S_j $(j \in \mathcal{I}_t)$, the receiver determines a linear combination $R(0) = 1/n \sum_{i=1}^{n} \omega_i$ and security condition $C_{4.1}$ is not guaranteed.

The polynomial interpolation-based unconditionally secure DOT protocol designed by Beimel et al. [1] assumes a semi-honest security model: the receiver may be curious but has to follow the protocol. Thus, in this model, security condition $C_{4.1}$ is guaranteed.

5 A More Robust Protocol

In [3], Blundo et al. have ameliorated the protocol presented in [2] to prevent (1) the servers and (2) the receiver from learning a linear combination of secrets. We show below that in spite of three different improvements, the protocol is still insecure regarding a greedy receiver.

5.1 First Improvement

To prevent servers from receiving a linear combination of secrets (see Sect. 4.1), each secret ω_i $(2 \leq i \leq n)$ is multiplicatively masked by an element r_i randomly selected in $\mathbb{K}$. More precisely, in the set-up phase, the sender $\mathcal{S}$ randomly selects $n - 1$ masks $r_2, r_3, \ldots, r_n$ in $\mathbb{K}$ and generates an $(n + 1)$-variate polynomial

$$Q(x, y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} U_i(x) \times V_i(y_1, y_2, \ldots, y_n),$$

where

- U_1 is a quasi-random polynomial of $\mathbb{K}_{k-1}[X]$ such that $U_1(0) = \omega_1$,
- U_i is a constant polynomial defined as $U_i(x) = r_i\omega_i - \omega_1$, for $i = 2, 3, \ldots, n$, and

[1] Because the first term is constant and public, it is not included in the request.

– V_i is an n-variate polynomial defined as $V_i(y_1, y_2, \ldots, y_n) = y_i$, for $i = 1, 2, \ldots, n$.

In the set-up phase, each server S_j ($j \in \mathcal{I}_m$) receives from the sender $\mathcal{S}$ an n-tuple $\boldsymbol{u}_j = (U_1(j), r_2\omega_2 - \omega_1, r_3\omega_3 - \omega_1, \ldots, r_n\omega_n - \omega_1)$, but also shares, generated by Shamir's secret sharing scheme [10], of $r_2, r_3, \ldots, r_n$.

In the oblivious transfer phase, the receiver $\mathcal{R}$ selects the index $\sigma \in [n]$ of the secret she wishes to obtain, as well as a set $\mathcal{I}_k \subset \mathcal{I}_m$ of k servers' indices. Then, she prepares an n-tuple $\boldsymbol{z}_j = (1, Z_2(j), Z_3(j), \ldots, Z_n(j))$ where Z_i ($2 \leq i \leq n$) is a quasi-random polynomial of $\mathbb{K}_{k-1}[X]$ such that $Z_i(0) = \delta^i_\sigma$. When a server S_j ($S_j \in \mathcal{I}_k$) receives a request $\boldsymbol{z}_j$, it calculates $\boldsymbol{v}_j = \boldsymbol{z}_j$ and returns to $\mathcal{R}$ not only $\boldsymbol{u}_j \bullet \boldsymbol{v}_j$ but also its shares of $r_2, r_3, \ldots, r_n$. From the collected k responses, $\mathcal{R}$ interpolates a polynomial R and calculates $r_\sigma\omega_\sigma = R(0)$ if $\sigma \neq 1$ or $\omega_1 = R(0)$ if $\sigma = 1$. If the former case, $\mathcal{R}$ also calculates r_σ from the collected shares and with a simple division the chosen secret, ω_σ.

We note that (1) the masks r_i ($2 \leq i \leq n$) may be nil since they are selected in $\mathbb{K}$ and that (2) ω_1 is not masked.

Thus, if $n = 2$, each server S_j ($j \in \mathcal{I}_m$) receives a pair $\boldsymbol{u}_j = (U_1(j), r_2\omega_2 - \omega_1)$ in the set-up phase. It is clear that if $\mathcal{R}$ wishes to obtain ω_2 and if $r_2 = 0$, after collecting k shares, she will determine $r_2\omega_2 = R(0) = 0$ and will be unable to calculate the value of ω_2. Therefore, the correctness (security condition C_1) of the protocol is not guaranteed; it follows that multiplicative masks $r_2, r_3, \ldots, r_n$ need to be selected in $\mathbb{K}^*$ and not in $\mathbb{K}$.

In addition, not masking ω_1 may provide the servers with information on ω_1 like shown in the following example.

Example 2. In the prime finite field $\mathbb{K} = \mathbb{F}_{11}$, we assume that $n = 2$ and that the probability distributions of ω_1 and ω_2 are:

$$\begin{cases} p_{\omega_1}(0) = 0.5, p_{\omega_1}(1) = 0.5, p_{\omega_1}(i) = 0 \text{ if } i \neq 0 \text{ and } i \neq 1, \\ p_{\omega_2}(1) = 0.5, p_{\omega_2}(3) = 0.5, p_{\omega_2}(i) = 0 \text{ if } i \neq 1 \text{ and } i \neq 3. \end{cases}$$

If the value received by the server S_j ($j \in \mathcal{I}_m$) in the set-up phase for $U_2(j) = r_2\omega_2 - \omega_1$ is 0, S_j is able to infer that $\omega_1 \neq 0$, hence $\omega_1 = 1$, because $r_2\omega_2 = \omega_1$, $r_2 \neq 0$ ($r_2 \in \mathbb{K}^*$ like shown above), $\omega_2 \neq 0$ ($\omega_2 = 1$ or $\omega_2 = 3$), and $\mathbb{K}$ is a field and consequently an integral domain.

It follows that in the sub-protocol presented by Blundo et al., the secret ω_1 should be masked like other secrets $\omega_2, \omega_3, \ldots, \omega_n$ and that all masks should be selected in $\mathbb{K}^*$.

We observe that if the use of masks is a good technique to prevent the servers from learning linear combinations of secrets, it does not change the situation of a greedy receiver, since she can determine all the masks. Therefore, once the protocol has been executed, the cheating receiver who has determined a linear combination of masked secrets easily obtains a linear combination of secrets, and from there, possibly all secrets (see Lemma 1). However, an advantage of

the masks is that the receiver cannot choose the coefficients of the linear combination of secrets she obtains by cheating. Indeed, each coefficient of the linear combination is the product of a coefficient chosen by the receiver with a mask, unknown from her at the time she prepares requests.

To conclude, the first improvement to Blundo et al.'s DOT protocol is insufficient to guarantee the sender's privacy. This is a contradiction with Blundo et al.'s claim ([3], Sect. 5, p. 350):

> "Notice that, in [8], for the case of two secrets, a proof that the Receiver can get *no more than a single* linear combination of the two secrets by running the sub-protocol described in Fig. 3 with k Servers was given. It is not difficult to show that the proof easily generalizes to our scheme for n secrets, i.e., after receiving information from k servers, the Receiver cannot learn more than a single linear combination of $\omega_1, \omega_2, \ldots, \omega_n$."

This is because, as stated by Lemma 1, the knowledge of a linear combination of secrets and of the probability distribution of the secrets may lead to the knowledge of all secrets.

5.2 Second Improvement

The major problem with the sub-protocol presented by Blundo et al. is that the receiver $\mathcal{R}$ is able to determine a linear combination of secrets, and then, depending on the probability distribution of secrets, the secrets themselves. However, if the secrets have a uniform distribution in the field $\mathbb{K}$, even if $\mathcal{R}$ (resp. a coalition of less than k servers) obtains a linear combination of secrets, she (resp. the coalition) cannot infer any of the secrets. So, the main idea underlying the second improvement is to transform a specific probability distribution into a uniform probability distribution. To this end, using the technique proposed by Naor and Pinkas [8], Blundo et al. have modified the sub-protocol so that it is executed twice: a first time on masks randomly selected in $\mathbb{K}$ (uniform distribution) and a second time on the products of the secrets and of the masks. To guarantee the consistency of receiver's requests, the same request sent by $\mathcal{R}$ to a server is used by the server for both the masks and the masked secrets.

The characteristics of the protocol are given in Annex A.3.

We observe that even if the protocol includes the technique suggested by Naor and Pinkas, the receiver may still obtain, not only a linear combination of secrets, but the secrets themselves (see example in Annex B).

In the demonstration proving that the protocol is secure, even with a greedy receiver (but with honest servers), Blundo et al. require an additional assumption: secrets cannot be identical. This assumption may be satisfied thanks to pads. For example, if q is a prime number such that $q \geq np$, the field $\mathbb{F}_p$ could be replaced with a field $\mathbb{F}_q$ and the pad $(i-1) \times p$ added to secret ω_i $(1 \leq i \leq n)$. We note that this additional assumption decreases the communication performance, since the new field has a cardinality larger than the original one.

However, even with this additional assumption, the claim on the security of the sender is incorrect (Condition (7) of Definition 2.2, p. 329 in [3], is not

satisfied) against a greedy receiver. The case is illustrated with the same example (Annex B), because according to the probability distributions chosen in the example, the secrets are necessarily different.

This is actually due to the first improvement which is not taken into account in Blundo et al.'s demonstration. Indeed, Naor and Pinkas demonstrated that the receiver could obtain the system of equations:

$$\begin{cases} \alpha_1 c_1 \omega_1 + \alpha_2 c_2 \omega_2 = R^{(1)}(0) \\ \alpha_1 c_1 + \alpha_2 c_2 = R^{(2)}(0) \end{cases}$$

where coefficients α_1 and α_2 are chosen by the receiver.

It is clear that if $R^{(2)}(0) = 0$, the receiver can infer $c_1 = \dfrac{-\alpha_2}{\alpha_1} c_2$ which, reported in the first equation gives $\alpha_2 c_2(\omega_2 - \omega_1) = R^{(1)}(0)$. Then, if $R^1(0) = 0$, then the receiver can infer the linear combination $\omega_2 - \omega_1 = 0$, because $c_2 \in \mathbb{K}^*$ and α_2 can be chosen different from 0 by the receiver). This explains the additional assumption.

However, in Blundo et al.'s protocol, each secret value is masked according to the first improvement (see Sect. 5.1). Therefore, in the case $n = 2$, the system of equations that the receiver is able to obtain is

$$\begin{cases} \alpha_1 r_1^{(1)} c_1 \omega_1 + \alpha_2 r_2^{(1)} c_2 \omega_2 = R^{(1)}(0) \\ \alpha_1 r_1^{(2)} c_1 + \alpha_2 r_2^{(2)} c_2 = R^{(2)}(0) \end{cases}$$

Again, if we assume that $R^{(2)}(0) = 0$, the receiver can infer $c_1 = \dfrac{-\alpha_2 r_2^{(2)}}{\alpha_1 r_1^{(2)}} c_2$ which, reported in the first equation gives $\alpha_2 c_2 (r_2^{(1)} \omega_2 - \dfrac{r_1^{(1)} r_2^{(2)}}{r_1^{(2)}} \omega_1) = R^{(1)}(0)$. If $R^{(1)}(0) = 0$, then the receiver can infer the linear combination $r_2^{(1)} r_1^{(2)} \omega_2 - r_1^{(1)} r_2^{(2)} \omega_1 = 0$. The coefficients $r_j^{(i)}$ $(i = 1, 2,\ 1 \leq j \leq n)$ being randomly selected in $\mathbb{K}^*$, there is no way to prevent the receiver from obtaining such a linear combination of the secrets.

However, it is easy to see that the first improvement is not useful when the second improvement is applied. Indeed, with the second improvement, servers do not receive linear combinations of secrets, but linear combination of masked secrets (which is the result of the first improvement) and linear combination of random masks. We conclude that only the second improvement of the protocol is necessary.

5.3 Third Improvement

Concerned with the degree of randomness necessary for the protocol, Blundo et al. suggest a simplification of the protocol to save a few random values ([3], Sect. 5, Remark p. 353):

> "However, we can show that *the same* random values $a_1, a_2, \ldots, a_{k-1}$ can be used in both instances of *SubDot(.)* and the values $r_2^{(2)}, r_3^{(2)}, \ldots, r_n^{(2)}$ can be computed as a function of $r_2^{(1)}, r_3^{(1)}, \ldots, r_n^{(1)}$."

We show in the following example, that sharing polynomials with the same coefficients $a_1, a_2, \ldots, a_{k-1}$ make the protocol insecure, regarding the sender's privacy.

Example 3. In the prime finite field $\mathbb{K} = \mathbb{F}_5$, we assume that $n = 2$ and that the probability distributions of ω_1 and ω_2 are:

$$\begin{cases} p_{\omega_1}(1) = 0.5, p_{\omega_1}(2) = 0.5, p_{\omega_1}(i) = 0 \text{ if } i \neq 1 \text{ and } i \neq 2, \\ p_{\omega_2}(0) = 0.5, p_{\omega_2}(4) = 0.5, p_{\omega_2}(i) = 0 \text{ if } i \neq 1 \text{ and } i \neq 4. \end{cases}$$

Since the secrets are masked only once (see previous section), we can also assume that

$$U_1^{(1)}(x) = c_1\omega_1 + \sum_{i=1}^{k-1} a_i x^i,$$

$$U_1^{(2)}(x) = c_1 + \sum_{i=1}^{k-1} a_i x^i,$$

$$U_2^{(1)}(x) = c_2\omega_2 - c_1\omega_1,$$

and

$$U_2^{(2)}(x) = c_2 - c_1$$

where $a_1, a_2, \ldots, a_{k-1}$ are randomly selected in $\mathbb{K}$.

In the set-up phase, each server S_j ($j \in \mathcal{I}_m$) receives from the sender $\mathcal{S}$ the pair $\boldsymbol{u}_j^{(1)} = (U_1^{(1)}(j), U_2^{(1)}(j))$ (for the masked secrets) as well as the pair $\boldsymbol{u}_j^{(2)} = (U_1^{(2)}(j), U_2^{(2)}(j))$ (for the masks). The server S_j is able to calculate $d_j = U_1^{(1)}(j) - U_1^{(2)}(j) = c_1(\omega_1 - 1)$. We assume that $d_j \neq 0$. Therefore, $\omega_1 - 1 \neq 0$ and so, $\omega_1 = 2$, according to the probability distribution p_{ω_1}. Hence, $c_1 = d_j/(\omega_1 - 1) = d_j$. Moreover, since S_j holds $c_2 - c_1$, the value of c_2 can be determined: $c_2 = (c_2 - c_1) + c_1 = (c_2 - c_1) + d_j$. The server S_j has received $c_2\omega_2 - c_1\omega_1$ from the sender and is able to determine c_1, c_2 and ω_1; To calculate ω_2 is easy. It follows that, given the probability distribution p_{ω_1} described above and assuming that $d_j \neq 0$, every server S_j is able to infer ω_1 and ω_2 in the set-up phase and the sender's privacy is not guaranteed (security condition C_3 is not satisfied).

This weakness extends to a greedy receiver and security condition $C_{4.1}$ could not be satisfied with this improvement. Indeed, if in the oblivious phase the receiver sends the request $\boldsymbol{z}_i = (1, 0)$ to $k-1$ servers S_i ($i \in \mathcal{I}_{k-1} \subset \mathcal{I}_k$, $|\mathcal{I}_{k-1}| = k-1$)

and $\boldsymbol{z}_\ell = (1, 1)$ to the k^{th} server S_ℓ ($\ell \in \mathcal{I}_k \setminus \mathcal{I}_{k-1}$), both secrets ω_1 and ω_2 can be determined thanks to the following method (we denote $P(x)$ the polynomial $\sum_{i=1}^{k-1} a_i x^i$):

- First, as soon as $\mathcal{R}$ has received from a server S_j ($j \in \mathcal{I}_{k-1}$) a response $(\boldsymbol{u}_j^{(1)} \bullet \boldsymbol{v}_j, \boldsymbol{u}_j^{(2)} \bullet \boldsymbol{v}_j)$, where $\boldsymbol{v}_j = \boldsymbol{z}_j$, she determines ω_1 with the technique described above. Thus, $\mathcal{R}$ receives $U_1^{(1)}(j) \times 1 + U_2^{(1)}(j) \times 0 = c_1\omega_1 + P(j)$ and $U_1^{(2)}(j) \times 1 + U_2^{(2)}(j) \times 0 = c_1 + P(j)$. She calculates $d_j = \boldsymbol{u}_j^{(1)} \bullet \boldsymbol{v}_j - \boldsymbol{u}_j^{(2)} \bullet \boldsymbol{v}_j = c_1(\omega_1 - 1)$. Because $d_j \neq 0$ and according to the probability distribution p_{ω_1}, $\mathcal{R}$ determines $\omega_1 = 2$. Hence, c_1 can be calculated too.
- Second, $\mathcal{R}$ determines $P(j)$ from the response of S_j: $P(j) = \boldsymbol{u}_j^{(1)} \bullet \boldsymbol{v}_j - c_1\omega_1$. The same operation can be executed from the responses of the other servers of $\mathcal{I}_{k-1}$, and $\mathcal{R}$ obtains $k-1$ values $P(j)$. The free coefficient of P is $P(0) = 0$. So, P may be written under the form $P = xP'$ where the degree of P' is at most $k-2$. With $k-1$ values $1/jP(j)$, the polynomial P' may be interpolated, which allows $\mathcal{R}$ to compute $P = xP'$.
- Finally, from the server S_ℓ, $\mathcal{R}$ obtains

$$\begin{aligned}(\boldsymbol{u}_\ell^{(1)} \bullet \boldsymbol{v}_\ell, \boldsymbol{u}_\ell^{(2)} \bullet \boldsymbol{v}_\ell) &= (c_1\omega_1 + P(\ell) + c_2\omega_2 - c_1\omega_1, c_1 + P(\ell) + c_2 - c_1) \\ &= (P(\ell) + c_2\omega_2, P(\ell) + c_2)\end{aligned}$$

  Since P has been computed in the second step, $\mathcal{R}$ is able to calculate c_2 from the second element of the pair and then the second secret, ω_2, from the first element of the pair.

This example shows that if the third improvement is applied, security condition $C_{4.1}$ is not satisfied either.

6 Conclusion

The main result of this paper is that when a party is able to obtain a linear combination of secrets, the sender's privacy (security conditions C_3 and $C_{4.1}$) may not be guaranteed for all secrets distributions. It follows that some weaknesses have been identified in the following polynomial interpolation-based DOT protocols:

- Naor and Pinkas's sparse polynomial interpolation-based protocol [8]: without the technique described in Sect. 4, p. 214, security conditions C_3 and $C_{4.1}$ are not guaranteed (in particular, Theorem 1 is incorrect). If the technique is applied, the size of the secrets space needs to be increased, and hence communication is less efficient (bigger shares need to be exchanged). The weakness is the same in Blundo et al.'s sparse polynomial interpolation-based protocol [2] which extends to n secrets Naor and Pinkas's protocol.

- Nikov et al.'s protocol [9] and Beimel et al.'s protocol [1]: for some values of the protocols' parameters, security condition C_3 is not guaranteed. However, this case is valid in regard to the security models presented by Nikov et al. and Beimel et al.: the protocols are considered as private even though each server holds the sender's secrets, assuming no server colludes with the receiver. On the other hand, Nikov et al.'s protocol [9] cannot guarantee security condition $C_{4.1}$, in spite of the claim of the designers.
- Blundo et al.'s protocol [3]: security condition $C_{4.1}$ is not guaranteed because of the combination of two techniques: the masking of secrets in the underlying sub-protocol and the parallel execution of the protocol on masked secrets and masks. If the masking of secrets in the underlying sub-protocol is removed, the protocol reaches the same level of security as Naor and Pinkas's protocol. In addition, the simplification suggested by Blundo et al. (reuse of the coefficients of the hiding polynomials) is a breach in the sender's security.

We also observe that the DOT protocol introduced by Cheong, Koshiba and Yoshiyama [5] is actually an application of the technique suggested by Naor and Pinkas to Nikov et al's DOT protocol; The level of security of the protocol is the same as in Naor and Pinkas's protocol.

A Characteristics of Some DOT Protocols

A.1 Naor and Pinkas's DOT [8]

In the sparse polynomial interpolation-based DOT protocol introduced by Naor and Pinkas [8], the number of secrets is $n = 2$, the threshold parameter is k, the sender's privacy and strong privacy parameters are $\lambda_S = k - 1$ and $\lambda_C = 0$, the hiding parameter is $N = 2$ and the encoding parameter is $e = 2$. The encoding function is $E(s) = \left(1, \delta_s^2\right)$ and the polynomials U_i and V_i $(1 \leq i \leq N)$ are:

- $U_1(x) = \omega_1 + \sum_{i=1}^{k-1} a_i x^i$, where coefficients a_i are randomly selected in $\mathbb{K}$, and $U_2(x) = \omega_2 - \omega_1$,
- $V_i(y_1, y_2) = y_i$ for $i = 1, 2$.

On the receiver's side, the number of contacted servers is $t = k$, the receiver's privacy parameter is $\lambda_R = k - 1$ and the first element of the encoding function being constant and public, it is not shared (i.e., $Z_1 = 1$) and is not included in the request transmitted by the receiver to the contacted servers.

A.2 Blundo Et Al.'s DOT [2]

The protocol introduced by Blundo, D'Arco, De Santis and Stinson [2] is an extension of Naor and Pinkas's sparse polynomial interpolation-based DOT protocol [8]. Only the following characteristics are different from those described in Appendix A.1.

– $n \geq 2$,
– $N = n$,
– $E(s) = \left(1, \delta_s^2, \delta_s^3, \ldots, \delta_s^n\right)$,
– $U_i(x) = \omega_i - \omega_1$ for $i = 2, 3, \ldots, N$,
– $V_i(y_1, y_2, \ldots, y_e) = y_i$ for $i = 1, 2, \ldots, N$.

A.3 Blundo Et Al.'s DOT [3]

The characteristics of the protocol introduced by Blundo, D'Arco, De Santis and Stinson [3] are similar to those of the protocol they presented in 2002 (see Appendix A.2). However, to improve the protocol, the secrets are masked twice:

– To prevent the servers from learning a linear combination of secrets, each secret ω_i $(2 \leq i \leq n)$ is multiplied by a mask r_i randomly selected in $\mathbb{K}$. Each mask is shared amongst the m servers involved in the protocol, thanks to Shamir's secret sharing schemes [10],
– To prevent the receiver from learning a linear combination of secrets, each secret ω_i $(1 \leq i \leq n)$ is masked with a mask c_i randomly selected in $\mathbb{K}^*$ and the receiver needs, with one request only, to collect shares of the chosen masked secret $c_\sigma \omega_\sigma$, but also of the corresponding mask c_σ.

More specifically, in the set-up phase, the sender $\mathcal{S}$ first selects masks c_i $(1 \leq i \leq n)$ in $\mathbb{K}^*$, which gives him two lists of secret values: $(c_1\omega_1, c_2\omega_2, \ldots, c_n\omega_n)$ and $(c_1, c_2, \ldots, c_n)$. Second, $\mathcal{S}$ selects random masks $r_i^{(1)}$ and $r_i^{(2)}$ $(2 \leq i \leq n)$ in $\mathbb{K}$ and builds two lists $L_1 = \left(c_1\omega_1, r_2^{(1)}c_2\omega_2, r_3^{(1)}c_3\omega_3, \ldots, r_n^{(1)}c_n\omega_n\right)$ and $L_2 = \left(c_1, r_2^{(2)}c_2, r_3^{(2)}c_3, \ldots, r_n^{(2)}c_n\right)$. Then, a set of N polynomials $U_i^{(1)}$ $(1 \leq i \leq N)$ is generated to hide the secrets values of L_1 and another set of of N polynomials $U_i^{(2)}$ $(1 \leq i \leq N)$ is generated to hide the secrets values of L_2:

– $U_1^{(1)}(x) = c_1\omega_1 + \sum_{i=1}^{k-1} a_i^{(1)} x^i$, where coefficients $a_i^{(1)}$ are randomly selected in $\mathbb{K}$, and for $i = 2, 3, \ldots, n$, $U_i^{(1)}(x) = r_i^{(1)}c_i\omega_i - c_1\omega_1$,
– $U_1^{(2)}(x) = c_1 + \sum_{i=1}^{k-1} a_i^{(2)} x^i$, where coefficients $a_i^{(2)}$ are randomly selected in $\mathbb{K}$, and for $i = 2, 3, \ldots, n$, $U_i^{(2)}(x) = r_i^{(2)}c_i - c_1$,

The e-variate polynomials V_i $(i = 1, 2, \ldots, N)$ are the same as those defined in Appendix A.2. Still in the set-up phase, each server S_j $(j \in \mathcal{I}_m)$ receives

$$\boldsymbol{u}_j^{(1)} = \left(U_1^{(1)}(j), r_2^{(1)}c_2\omega_2 - c_1\omega_1, r_3^{(1)}c_3\omega_3 - c_1\omega_1, \ldots, r_n^{(1)}c_n\omega_n - c_1\omega_1\right)$$

and

$$\boldsymbol{u}_j^{(2)} = \left(U_1^{(2)}(j), r_2^{(2)}c_2 - c_1, r_3^{(2)}c_3 - c_1\omega_1, \ldots, r_n^{(2)}c_n - c_1\right),$$

as well as the shares $[r_2^{(1)}]_j, [r_3^{(1)}]_j, \ldots, [r_n^{(1)}]_j$ and $[r_2^{(2)}]_j, [r_3^{(2)}]_j, \ldots, [r_n^{(2)}]_j$ (If $F_t^{(s)}$ is the hiding polynomial determined in Shamir's secret sharing scheme to share $r_t^{(s)}$, the share $F_t^{(s)}(j)$ allocated to server S_j is denoted $[r_t^{(s)}]_j$.)

In the oblivious phase, on reception of the request $\boldsymbol{z}_j$, a server S_j ($j \in \mathcal{I}$) calculates $\boldsymbol{v}_j = (V_1(\boldsymbol{z}_j), V_2(\boldsymbol{z}_j), \ldots, V_N(\boldsymbol{z}_j))$ and returns $\boldsymbol{u}_j^1 \bullet \boldsymbol{v}_j$ and $\boldsymbol{u}_j^2 \bullet \boldsymbol{v}_j$, with the two sets of $n-1$ shares of $\left(r_2^{(1)}, r_3^{(1)}, \ldots, r_n^{(1)}\right)$ and $\left(r_2^{(2)}, r_3^{(2)}, \ldots, r_n^{(2)}\right)$ to the receiver. From the collected values, $\mathcal{R}$ interpolates two polynomials $R^{(1)}$ and $R^{(2)}$ and, if $\sigma = 1$, calculates $R^{(1)}(0) = c_\sigma \omega_\sigma$ and $R^{(2)}(0) = c_\sigma$. If $\sigma \neq 1$, $\mathcal{R}$ calculates $R^{(1)}(0) = c_\sigma r_\sigma^{(1)} \omega_\sigma$ and $R^{(2)}(0) = c_\sigma r_\sigma^{(2)}$ and also determines from the k collected shares $[r_\sigma^{(1)}]_j$ the value of $r_\sigma^{(1)}$ and similarly, from the k collected shares $[r_\sigma^{(2)}]_j$ the value of $r_\sigma^{(2)}$. Then, with simple division(s), $\mathcal{R}$ determines c_σ first and ω_σ second.

A.4 Nikov Et Al.'s DOT [9]

The sparse polynomial interpolation-based DOT protocol introduced by Nikov, Nikova, Preneel and Vandewalle [9] is characterized by the following parameters: the number of secrets $n \geq 2$, the threshold parameter k, the sender's privacy parameter $\lambda_S \leq k-1$, the receiver's privacy parameter $\lambda_R \leq k-1$, the hiding parameter $N = n$ and the encoding parameter $e = n$. The parameter λ_C is defined such that $\lambda_R + \lambda_C \leq k-1$. In addition, the encoding function is $E(s) = \left(1, \delta_s^2, \delta_s^3, \ldots, \delta_s^n\right)$ and the polynomials U_i and V_i ($1 \leq i \leq N$) are:

- $U_1(x) = \omega_1 + \sum_{\ell=1}^{k-1} a_{1,\ell} x^\ell$, where coefficients $a_{1,\ell}$ are randomly selected in $\mathbb{K}$, and for $i = 2, 3, \ldots, N$, $U_i(x) = \omega_i - \omega_1 + \sum_{\ell=1}^{\lambda_C} a_{i,\ell} x^\ell$, where coefficients $a_{i,\ell}$ are randomly selected in $\mathbb{K}$,
- $V_i(y_1, y_2, \ldots, y_e) = y_i$ for $i = 1, 2, \ldots, N$.

Like in Naor and Pinkas's and in Blundo et al.'s DOT protocols (see Appendices A.1 and A.2 above), on the receiver's side, the number of contacted servers is $t = k$ and the first element of the encoding function being constant and public, it is not shared (i.e., $Z_1 = 1$) and is not included in the request transmitted by the receiver to the contacted servers.

A.5 Beimel Et Al.'s DOT [1]

In [1], Beimel, Chee, Wang and Zhang propose a specific reduction from a DOT protocol to a polynomial interpolation-based information-theoretic private information retrieval protocol. The characteristics of the protocol are: the number of secrets $n \geq 2$, the threshold parameter k, the sender's privacy and strong privacy parameters $\lambda_S = \lambda_C \leq k-1$, the receiver's privacy parameter $\lambda_R \leq k-1$, the hiding parameter $N = n+1$ and the encoding parameter $e > 0$. The polynomials U_i and V_i ($1 \leq i \leq N$) are:

- $U_1(x) = \sum_{i=0}^{k-1} a_{1,i} x^i$, where coefficients $a_{1,i}$ are randomly selected in $\mathbb{K}$, and for $i = 2, 3, \ldots, N$, the polynomial U_i is defined by $U_i(x) = (\omega_{i-1} - a_{1,0}) + \sum_{j=1}^{\lambda_C} a_{i,j} x^j$, where coefficients $a_{i,j}$ are randomly selected in $\mathbb{K}$,

– $V_1(y_1, y_2, \ldots, y_e) = 1$ and for $i = 2, 3, \ldots, N$, the polynomial V_i and the encoding function E must satisfy $V_i(E(\ell)) = \delta_\ell^{i-1}$ for $\ell \in [n]$.

On the receiver's side, the number of contacted servers is $t = k$. In addition, for efficiency purposes, each contacted server S_j ($j \in \mathcal{I}_t$) transforms the share $\boldsymbol{u}_j \bullet \boldsymbol{v}_j$ into a split s_j, which is sent back to the receiver. The receiver has just to calculate the sum $\omega_\sigma = \sum_{j \in \mathcal{I}_t} s_j$ to obtain the chosen secret.

B Example of Insecurity in Blundo et al.'s DOT Protocol

—— **Public Information** ——

– Finite field $\mathbb{F}_{11}$
– Threshold $k = 3$
– Number of secrets $n = 2$
– $p_{\omega_1}(6) = 0.5$, $p_{\omega_1}(2) = 0.5$, $p_{\omega_1}(i) = 0$ if $i \neq 6$ and $i \neq 2$
– $p_{\omega_2}(1) = 0.5$, $p_{\omega_2}(3) = 0.5$, $p_{\omega_2}(i) = 0$ if $i \neq 1$ and $i \neq 3$

—— **Set-up phase** ——

Information private to the sender:

– $\omega_1 = 6$ and $\omega_2 = 1$
– $c_1 = 5$ and $c_2 = 7$
– $r_1^{(1)} = 1$ and $r_2^{(1)} = 2$
– $r_1^{(2)} = 4$ and $r_2^{(2)} = 5$

Intermediate calculus to prepare the sharing polynomials:

– $r_1^{(1)} \times c_1 \times \omega_1 = 8$
– $r_2^{(1)} \times c_2 \times \omega_2 = 3$
– $r_1^{(2)} \times c_1 = 9$
– $r_2^{(2)} \times c_2 = 2$
– Sharing polynomial $U_1^{(1)} = 8 + 2X + 9X^2$
– Sharing polynomial $U_1^{(2)} = 9 + X + 4X^2$
– S_1 receives $\boldsymbol{u}_1^{(1)} = (8, 6)$ and $\boldsymbol{u}_1^{(2)} = (3, 4)$
– S_2 receives $\boldsymbol{u}_2^{(1)} = (4, 6)$ and $\boldsymbol{u}_2^{(2)} = (5, 4)$
– S_3 receives $\boldsymbol{u}_3^{(1)} = (7, 6)$ and $\boldsymbol{u}_3^{(2)} = (4, 4)$

—— **Transfer phase** ——

– Request generated by the receiver: $\boldsymbol{z}_j = (1, 6)$
– $\boldsymbol{v}_j = \boldsymbol{z}_j = (1, 6)$
– S_1 replies with $\boldsymbol{u}_1^{(1)} \bullet \boldsymbol{v}_1 = 0$ and $\boldsymbol{u}_1^{(2)} \bullet \boldsymbol{v}_1 = 5$
– S_2 replies with $\boldsymbol{u}_2^{(1)} \bullet \boldsymbol{v}_2 = 7$ and $\boldsymbol{u}_2^{(2)} \bullet \boldsymbol{v}_2 = 7$
– S_3 replies with $\boldsymbol{u}_3^{(1)} \bullet \boldsymbol{v}_3 = 10$ and $\boldsymbol{u}_3^{(2)} \bullet \boldsymbol{v}_3 = 6$

—— **The receiver tries to obtain all secrets** ——

- Interpolated polynomal from $(1,0)$, $(2,7)$, and $(3,10)$: $R^{(1)} = 2X + 9X^2$
- $r_2^{(1)} c_2\omega_2 + r_1^{(1)} c_1\omega_1 = 2 \times R^{(1)}(0) = 0$
- Interpolated polynomal from $(1,5)$, $(2,7)$, and $(3,6)$: $R^{(2)} = X + 4X^2$
- $r_2^{(2)} c_2 + r_1^{(2)} c_1 = 2 \times R^{(2)}(0) = 0$
- From the received mask shares, the receiver determines $r_1^{(1)} = 1$, $r_2^{(1)} = 2$, $r_1^{(2)} = 4$ and $r_2^{(2)} = 5$.

So, the receiver can infer the two equations:

$$\begin{cases} 2 \times c_2\omega_2 + 1 \times c_1\omega_1 = 0 \\ 5 \times c_2 + 4 \times c_1 = 0 \end{cases}$$

From the second equation, the receiver infers that $c_2 = 8 \times c_1$. Reporting the equality in the first equation, she obtains: $c_1 \times (5 \times \omega_2 + 1 \times \omega_1) = 0$. If $(\omega_1, \omega_2) =$

- $(6,1)$, then $5 \times \omega_2 + 1 \times \omega_1 = 0$,
- $(2,1)$, then $5 \times \omega_2 + 1 \times \omega_1 = 7$,
- $(6,3)$, then $5 \times \omega_2 + 1 \times \omega_1 = 10$,
- $(2,3)$, then $5 \times \omega_2 + 1 \times \omega_1 = 6$.

The only pair of secrets which satisfies the first equation is: $(6,1)$. The greedy receiver has obtained all secrets.

References

1. Beimel, A., Chee, Y.M., Wang, H., Zhang, L.F.: Communication-efficient distributed oblivious transfer. J. Comput. Syst. Sci. **78**(4), 1142–1157 (2012)
2. Blundo, C., D'Arco, P., De Santis, A., Stinson, D.R.: New results on unconditionally secure distributed oblivious transfer (extended abstract). In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 291–309. Springer, Heidelberg (2003)
3. Blundo, C., D'Arco, P., De Santis, A., Stinson, D.R.: On unconditionally secure distributed oblivious transfer. J. Cryptol. **20**(3), 323–373 (2007)
4. Brassard, G., Crépeau, C., Robert, J.M.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
5. Cheong, K.Y., Koshiba, T., Nishiyama, S.: Strengthening the security of distributed oblivious transfer. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 377–388. Springer, Heidelberg (2009)
6. Friedman, W.F.: The index of coincidence and its applications in cryptography. No. 22 in Riverbank Publications, Riverbank Laboratories, Geneva, IL, USA (1922)
7. Kasiski, F.W.: Die Geheimschriften und die Dechiffrir-Kunst. Mittler & Sohn, Berlin (1863)
8. Naor, M., Pinkas, B.: Distributed oblivious transfer. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 205–219. Springer, Heidelberg (2000)
9. Nikov, V., Nikova, S., Preneel, B., Vandewalle, J.: On unconditionally secure distributed oblivious transfer. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 395–408. Springer, Heidelberg (2002)
10. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)