Exploratory Designs of Unconditionally Secure Distributed Oblivious Transfer Protocols

Thesis submitted by

Christian L. F. CORNIAUX
BEng(Mech) *E.S.E.M.*, MSc(IT) *E.N.S.M.S.E.*

January 2016

for the degree of Doctor of Philosophy
in the College of Business, Law & Governance, Information Technology Academic Group
James Cook University

To $\mathbb{C}^3$. May the Force be with you.

List of Publications and Contribution of Others

The articles hereafter are based on chapters of this thesis. Each article was written under the supervision of Hossein Ghodosi who provided editorial and academic advice, as well as scrupulous and critical reviews. In addition, the articles were peer-reviewed and presented at international conferences in the fields of cryptography and information security.

[1] C. L. F. Corniaux and H. Ghodosi. 'Scalar Product-Based Distributed Oblivious Transfer'. In: *Information Security and Cryptology - ICISC 2010*. Ed. by K.-H. Rhee and D. Nyang. Vol. 6829. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2011, pp. 338–354.
Article based on Chap. 4.

[2] C. L. F. Corniaux and H. Ghodosi. 'A Verifiable Distributed Oblivious Transfer Protocol'. In: *Information Security and Privacy - ACISP 2011*. Ed. by U. Parampalli and P. Hawkes. Vol. 6812. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2011, pp. 444–450.
Article based on Chap. 5.

[3] C. L. F. Corniaux and H. Ghodosi. 'T-out-of-n Distributed Oblivious Transfer Protocols in Non-adaptive and Adaptive Settings'. In: *Information Security Practice and Experience - ISPEC 2012*. Ed. by M. D. Ryan, B. Smyth, and G. Wang. Vol. 7232. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2012, pp. 126–143.
Article based on Chaps. 6 and 7.

[4] C. L. F. Corniaux and H. Ghodosi. 'An Information-Theoretically Secure Threshold Distributed Oblivious Transfer Protocol'. In: *Information Security and Cryptology - ICISC 2012*. Ed. by T. Kwon, M.-K. Lee, and D. Kwon. Vol. 7839. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2013, pp. 184–201.
Article based on Chap. 8.

[5] C. L. F. Corniaux and H. Ghodosi. 'An Entropy-based Demonstration of Shamir's Secret Sharing Scheme'. In: *International Conference on Information Science, Electronics and Electrical Engineering - ISEEE 2014*. Ed. by X. Jiang, S. Li, Y. Dai, and Y. Cheng. IEEE, 2014, pp. 46–48.
Article based on App. A.

[6] C. L. F. Corniaux and H. Ghodosi. 'Security Analysis of Polynomial Interpolation-based Distributed Oblivious Transfer Protocols'. In: *Information Security and Cryptology - ICISC 2014*. Ed. by J. Lee and J. Kim. Vol. 8949. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2015, pp. 363–380.
Article based on Chap. 3.

Professional editors, Jean and Alan Dartnall, provided copy-editing and proofreading services for this thesis, according to the guidelines laid out in the university-endorsed national 'Guidelines for editing research theses'. These services were partially funded by the College of Business, Law & Governance of James Cook University.

More globally, in accordance with James Cook University's Minimum Resources Policy, I received financial assistance ($4,500) for my PhD research from the School of Business (Faculty of Law, Business and the Creative Arts). In addition, attendance at conferences (around $6,000) was funded via the Graduate Research Scheme.

Acknowledgements

First of all, I would like to thank my principal supervisor, Hossein Ghodosi, for his stimulating guidance, advice, and support during my PhD studies at James Cook University. In 2007, his excellent postgraduate course CP5110 made me realize that cryptography is at the confluence of mathematics, engineering and computer science, three domains of which I am fond. I have enjoyed working with Hossein and appreciated above all his technical rigour as well as his optimism for finding solutions.

I also wish to express my gratitude to the other members of my advisory team, Bruce Litow and Ahmad Zahedi, who have kindly watched my progress from afar.

I am also grateful to the officers of the Graduate Research School and of the Faculty of Law, Business and the Creative Arts (now College of Business, Law & Governance) who managed my candidature; in particular, I am indebted to Erica O'Sullivan, Janie Edwards and Michelle Morisson who helped me with the organization of conference travels. It was almost an easy task!

Finally, and most importantly, I would like to thank my wife who accepted that I devoted a few years to cryptographic research, probably to the detriment of our family life. I am sure she will miss all the metaphors I made up to explain my findings...

Abstract

The security of digital goods buyers and sellers is unbalanced. Of course, the property of sellers is protected; for example, when customers acquire digital books or films from Internet's merchants, they only receive the products they have paid for. Unfortunately, the buyers' privacy is rarely respected: purchases are often — without the buyers' knowledge — monitored, recorded, analysed, and sometimes sold to marketing companies. As a consequence, even if the customers do not intend to acquire additional products, their computer screens are later invaded with targeted advertisements.

The main purpose of this thesis is to propose some methods to restore the balance and guarantee the buyers' privacy, while protecting the interests of the sellers.

To this end, it is worth looking into the area of cryptography and more specifically, it is worth studying and designing some protocols called *distributed oblivious transfer* (DOT) protocols. A DOT protocol allows a party $\mathcal{A}$ to obtain one of the secret pieces of information (a *secret* for short) held by a party $\mathcal{B}$ so that the following two fundamental conditions are satisfied:

- $\mathcal{A}$ chooses the secrets she wishes to obtain and does not obtain anything on the secrets she has not chosen,
- $\mathcal{B}$ does not learn which secret was obtained by $\mathcal{A}$.

Furthermore, to improve the availability of the information, the protocol is distributed. That is, the party $\mathcal{B}$ transmits his secret information to m servers and the party $\mathcal{A}$ needs to contact at least k of these servers to obtain the chosen secret. The servers are not fully trusted, neither by $\mathcal{A}$, nor by $\mathcal{B}$. Therefore, from the information exchanged with $\mathcal{A}$ and $\mathcal{B}$, no coalition of servers should be able to learn the secrets of $\mathcal{B}$ or the choice of $\mathcal{A}$.

The results of a preliminary literature review are surprising. In fact, the number of publications on DOT protocols is small (fewer than 20) compared to, for example, the number of publications on a similar concept, secret sharing (100s of publications). And yet, oblivious transfer is a fundamental component of more complex cryptographic protocols such as multi-party computation protocols, which allow a group of participants to securely calculate any function of their joint secret inputs. So, one could expect many variants, for example of the original DOT protocol introduced in 2002 by Naor and Pinkas [74], to fulfil the requirements of specific scenarios.

The design of variants of DOT protocols in traditional cryptography has been the guiding thread of my research. My contribution mainly consists in (*a*) a critical analysis of the existing protocols, demonstrating their limitations, weaknesses, and in some cases, flaws; and (*b*) the design of the following protocols, well adapted to some specific situations:

A Strongly Secure DOT Protocol. This DOT protocol addresses the most important weakness of unconditionally secure, one-round, polynomial interpolation-based DOT protocols: after the protocol has been executed, if the party $\mathcal{A}$ corrupts only one server, she can learn all the secrets of the party $\mathcal{B}$. The protocol is secure even if $\mathcal{A}$ corrupts up to $k-1$ servers.

A Verifiable DOT Protocol. The party $\mathcal{A}$ should obtain the secret she has chosen, even when some servers are controlled by a malicious adversary whose objective is to sabotage the protocol. This is the case with this protocol, assuming that the adversary cannot control more than $k-1$ servers.

A Multiple Secrets DOT Protocol. When the party $\mathcal{A}$ wishes to obtain $n > 1$ secrets, the current protocols have to be executed n times. In this context, they are inefficient. The DOT protocol introduced here allows the party $\mathcal{A}$, by contacting in the same session $k-1+n$ servers, to collect n secrets.

Adaptive DOT Protocols. The previous protocol allows the party $\mathcal{A}$ to request several secrets. However, the request of one secret may depend on the values of secrets already obtained. Two efficient protocols are presented in this scenario. The first one allows $\mathcal{A}$ to receive a limited number of secrets and therefore, is well adapted to a single receiver. For several receivers, a second protocol is proposed. This second protocol accepts an unlimited number of queries, but requires communications amongst the servers.

A Threshold DOT protocol. Most existing DOT protocols rely on threshold secret sharing schemes. In a k-threshold protocol or scheme, security is guaranteed not only

when k parties are contacted, but also when more than k parties are contacted. However, the existing DOT protocols based on k-threshold secret sharing schemes require an additional mechanism to control that exactly k servers are contacted, which is an under-utilisation of the underlying functionality. The proposed protocol is the first k-threshold DOT protocol which allows the party $\mathcal{A}$ to contact as many servers as she wishes to obtain the chosen secret, provided that at least k servers are contacted.

This research is limited to unconditionally secure protocols, i.e., protocols whose security does not depend on mathematical (unproven) assumptions; within the limits of the given security models, the protocols are secure even against an adversary with unlimited computing power and time.

In brief, the results presented in this thesis are a significant advance to the state of the art research on DOT protocols because on one hand, they point out the weaknesses of the DOT protocols most commonly accepted by the cryptographic community and on the other hand, they contribute to the cryptographic field through the design of new protocols, secure and efficient.

Contents

List of Figures

List of Tables

Notations

$\mathbb{N}$	Set $\{ 0, 1, \dots \}$ of natural numbers
$\mathbb{Z}$	Set $\{ \dots, -1, 0, 1, \dots \}$ of integers
$\mathbb{Q}$	Set of rational numbers
$\mathbb{R}^+$	Set of non-negative real numbers
$\mathbb{G}^*$	Assuming $[\, \mathbb{G}, + \,]$ is a group, $\mathbb{G} \setminus \{ 0 \}$
$\mathrm{GF}(p)$	Galois field ($p \in \mathbb{N}$, p power prime)
$\mathcal{M}_{k,\ell}(\mathbb{G})$	Group of matrices (k rows, ℓ columns) with coefficients in $\mathbb{G}$
$\mathfrak{S}_n$	Group of cyclic permutations on a set of n elements
$\lfloor x \rfloor$	The greatest integer smaller or equal to x
$\lceil x \rceil$	The smaller integer greater or equal to x
δ_i^j	Kronecker symbol ($\delta_i^j = 1$ if $i = j$ and $\delta_i^j = 0$ otherwise)
$\lvert S \rvert$	Cardinality of the set S
S^n	The set $S \times S \times \cdots \times S$ (n times)
$\mathcal{P}(S)$	The power set of S
$[\![n_1, n_2]\!]$	Set of natural numbers $\{ n_1, n_1 + 1, \dots, n_2 \}$
$\mathbb{K}[X]$	Ring of univariate polynomials with coefficients in $\mathbb{K}$
$\mathbb{K}[Y_1, Y_2, \dots, Y_n]$	Ring of n-variate polynomials with coefficients in $\mathbb{K}$
$\mathbb{K}_k[X]$	Group of univariate polynomials of degree at most k, where $k \in \mathbb{N}$
$\log_2(x)$	Logarithm function in base 2 of x, where $x \in \mathbb{R}^+ \setminus \{ 0 \}$
$b_1 \oplus b_2$	XOR ('exclusive or') operation between the bits b_1 and b_2
$S_1 \subseteq S_2$	S_1 is a subset of S_2
$S_1 \subset S_2$	$S_1 \subseteq S_2$ and $S_1 \neq S_2$

$\varnothing$	The empty set
$S_1 \cap S_2$	Intersection of the sets S_1 and S_2
$S_1 \cup S_2$	Union of the sets S_1 and S_2
$S_1 \setminus S_2$	Set difference (set of elements belonging to S_1 but not to S_2)
Pr	Probability measure on an event space $\Sigma \subseteq \mathcal{P}(\Omega)$, where Ω is a sample space
$x \in S$	Membership of a set (x is an element of the set S)
$p_{\boldsymbol{X}}$	Probability mass function of the random variable $\boldsymbol{X}$
$x \in_R S$	The element x is randomly selected in the non-empty finite set S, i.e., for all $a \in S$, $p_{\boldsymbol{X}}(a) = 1/\lvert S\rvert$
$\boldsymbol{u} = [\, u_1, u_2, \ldots, u_n \,]$	If $n > 0$, n-tuple $\boldsymbol{u}$ (bold font), i.e., sequence of elements $u_1, u_2, \ldots, u_n$
$\boldsymbol{m}_n$	If $n \in \mathbb{N}$, n-tuple $[\, m, m, \ldots, m \,]$
$\boldsymbol{u} \bullet \boldsymbol{v}$	If $\boldsymbol{u} = [\, u_1, u_2, \ldots, u_n \,]$ and $\boldsymbol{v} = [\, v_1, v_2, \ldots, v_n \,]$ ($n > 0$), then $\boldsymbol{u} \bullet \boldsymbol{v} = \sum_{i=1}^{n} u_i \times v_i$
$\deg(P)$	Degree of the polynomial P
$n!$	Factorial of $n \in \mathbb{N}$, i.e., $1 \times 2 \times \cdots \times (n-1) \times n$ (by convention, $0! = 1$)
$\binom{n}{p}$	Binomial coefficient: $\binom{n}{p} = \frac{n!}{p!(n-p)!}$
$O(f(x))$	Limiting behaviour of the function f ($g(x) \in O(f(x))$ if there exists $c > 0$ and x_0 such that $g(x) < cf(x)$ for $x > x_0$)

1

Introduction

Today, users who purchase products on the Internet frequently have their requests tracked and analysed by marketers; later, numerous advertisements related to the requests 'magically' appear on their computers' screens. Without doubt, more privacy would be welcome. On the other hand, when the same users buy and download, for example, digital books or films, the sellers do not wish the users to freely access products which were not paid for. It is clear that a double protection should be offered: the privacy of the users should be maintained and the security of the sellers should be guaranteed. This is where cryptography can help, and especially the cryptographic functionality called *distributed oblivious transfer*.

Indeed, if etymologically, cryptography (from ancient Greek κρυπτός, meaning *private*, *hidden* and γράφειν, meaning *to write*) is the study of what is hidden, by extension, it may also be considered as the science of controlling the disclosure of information. This is the purpose of most cryptographic protocols, including *distributed oblivious transfer* protocols; a distributed oblivious transfer protocol allows a party $\mathcal{A}$ to obtain one of the pieces of information held by a party $\mathcal{B}$ while the security of $\mathcal{B}$ and the privacy of $\mathcal{A}$ are preserved. Furthermore, the availability of the information is improved thanks to a distributed environment. Thus, the request of $\mathcal{A}$ is sent to 'many' intermediate parties who hold only parts of the original information. Then, as soon as $\mathcal{A}$ has received 'enough' responses, the requested information may be determined. Of course, 'many' and 'enough' need to be specified more formally.

In the literature, few articles describing distributed oblivious transfer protocols have been published, whereas some similar protocols, like secret sharing protocols, have been thoroughly studied during the last 35 years. Yet, both types of protocols are fundamental and may be applied in similar scenarios. This lack of research

has motivated further investigation. In this thesis, the existing distributed oblivious transfer protocols are analysed and novel ones, particularly well suited to some specific scenarios, are designed.

The remainder of this chapter is structured as follows. In Sect. 1.1 is refined the concept of distributed oblivious transfer and is introduced the concept of unconditional security. Then, Sect. 1.2 gives a brief description of related cryptographic areas (oblivious transfer, private information retrieval and multi-party computation). In the following section (Sect. 1.3) is presented a short literature survey of distributed oblivious transfer protocols. Section 1.4 specifies the objectives and limitation of the research. Finally, the main contributions are presented and an overview of the thesis organization is given in Sect. 1.5.

1.1 Definitions

Unconditionally secure distributed oblivious transfer is the overarching concept on which this thesis is based. This is why, first and foremost, the definition of distributed oblivious transfer is introduced in this section. Second, the different categories of cryptographic security are described and, more precisely, unconditional security is defined.

1.1.1 Distributed Oblivious Transfer

Informally, *distributed oblivious transfer* (DOT) protocols allow a party, the *receiver*, to obtain one of the secrets held by another party, the *sender*, by collecting information from several third parties, the *servers*. To follow the cryptographic tradition and to set a concrete image of DOT protocols' participants, the receiver is considered as a person (Alice, or $\mathcal{R}$ in this document) as well as the sender (Bob, or $\mathcal{S}$ in this document). The third parties, the servers, may be considered as machines having memory and processing power, able to communicate with the sender and the receiver.

The main specificities of a DOT protocol are that, once the protocol has been executed, on one hand the sender does not know which secret was obtained by the receiver and on the other hand, the receiver has no information on the secrets she did not choose. In addition, it is assumed that the servers cannot be trusted; that is, some of them could be controlled by the sender, the receiver or even an external adversary. Therefore, the protocol should guarantee that a coalition between the sender (or an adversary) and some servers cannot learn the receiver's choice. Similarly, a coalition between an adversary and some servers should not be able to learn any of the sender's secrets. Likewise, a coalition between the receiver and some servers should not be able to learn any of the sender's secrets before the

protocol is executed and none of the secrets the receiver did not obtain after the protocol has been executed.

A key characteristics of DOT protocols is the size of the coalitions: researchers try to design DOT protocols where secret information (the sender's secrets and the receiver's choice) are kept secret against coalitions as large as possible.

1.1.2 Cryptographic Security Typology

In a conventional cryptosystem, a clear message is encrypted thanks to a key $k_{\mathcal{A}}$ by a party $\mathcal{A}$, who transmits the resulting cryptogram to a party $\mathcal{B}$. Thanks to a decrypting key $k_{\mathcal{B}}$ (possibly $k_{\mathcal{A}} = k_{\mathcal{B}}$), $\mathcal{B}$ is then able to decrypt the cryptogram and obtain the original message. It is assumed that an adversary is able to obtain the cryptogram and, without any key, tries to determine the original message. The security of such a cryptosystem falls in one of the two categories:

Complexity-Theoretic Security. The adversary is able, by conducting an exhaustive search in the key space, to determine the original message. However, the cryptosystem is designed so that (*a*) there is no other method than the exhaustive search to find out the clear message and (*b*) the key space size is specified so that an exhaustive search is impractical in a limited time (insufficient processing power regarding the number of operations to execute or lack of memory to store intermediate results). Of course the parameters of the cryptosystem need to be frequently adjusted to take into account the increasing efficiency of computers. Since the cryptosystem's security depends on the available computing resource, it is also called *computational security*.

Today, the security of computationally secure cryptosystems relies on mathematical conjectures in number theory. For example, the security of all public-key cryptosystems is based on the assumed hardness of computational problems such as the computing of discrete logarithms in particular finite cyclic groups (e.g., Diffie-Hellman's cryptosystem [44]), the factorization of integers (e.g., RSA cryptosystem [86]), the computing of square roots in finite fields (e.g., Rabin's cryptosystem [80]), etc.

Note that if a conjecture is proved (now a theorem), an adversary can still theoretically breach the security of the cryptosystem based on this conjecture, using an exhaustive search. However, if the conjecture is disproved (no longer a conjecture), the cryptosystem's security is not guaranteed any more.

Information-Theoretic Security. Even with unlimited computing processing power, unlimited time, and infinite storage capability, an adversary cannot break the security

of an information-theoretically secure cryptosystem. The cryptosystem's security can be proved thanks to information-theoretic tools [90–92] and therefore cannot depend on any unproven mathematical condition. Such a cryptosystem, not relying on any mathematical conjecture, is also said to be *unconditionally secure* [44].

By extension, cryptographic protocols also fall in the computational or unconditional security categories; this is the case for DOT protocols. The protocols designed in this thesis are all unconditionally secure, which allows the demonstration of their security. Furthermore, to abide by the cryptographic tradition, the design of the DOT protocols follows Kerckhoffs' principle, i.e., the details of the protocols are public whereas only the private inputs need to be secret [67, 68].

1.2 Related Areas

In this section are briefly described cryptographic classes of protocols, with strong links to DOT protocols. More specifically, the *oblivious transfer*, *private information retrieval* and *multi-party computation* protocols are reviewed. For each of these three classes, it is shown how to reduce a DOT protocol to a protocol of the class.

1.2.1 Oblivious Transfer

An *oblivious transfer* (OT) protocol allows two parties to exchange, in total privacy, one or more secret messages. The first OT protocol, introduced by Rabin in 1981 [81], enables a sender to transmit a message to a receiver in such a way that the receiver gets the message with probability $1/2$ while the sender does not know whether the message was received. A similar concept, presented under the form of a quantum multiplexing channel-based primitive, had already been introduced by Wiesner in years 1970's, as reported by Brassard and Crépeau [23]. However, Wiesner's first formal publication describing the novel primitive was not published until 1983 [97]. This primitive allows a party $\mathcal{B}$ to send two messages to a party $\mathcal{A}$ in a way that (*a*) $\mathcal{A}$ can decide which message to read and (*b*) the other message is irreversibly destroyed.

In 1985, Even, Goldreich, and Lempel [47] introduced a variant of Rabin's OT protocol for a contract signature application. This OT protocol, identified as OT-$\binom{2}{1}$, is an exchange protocol between a receiver and a sender who has two secret messages; the receiver chooses one of the two messages and the sender transmits the chosen message to the receiver. At the end of the protocol, the sender does not know which message was selected and the receiver knows nothing of the other message. The OT-$\binom{2}{1}$ was then generalized to a 1-out-of-n

OT protocol — or OT-$\binom{n}{1}$ — by Brassard, Crépeau, and Roberts [24] in 1987 under the name *all-or-nothing disclosure of secrets* (to be understood as 'one and only one secret disclosure', also designated sometimes by the acronym *ANDOS*).

It was later demonstrated that all these variants of OT protocols are equivalent, whether the secret messages are bits or strings of bits [25, 26, 36, 39]. More generally, OT protocols are fundamental components of secure distributed computation [61, 69]. They can be used in the design of oblivious function evaluation protocols, which enable two parties to evaluate a function of their inputs without revealing the inputs themselves. Therefore, numerous OT protocol variants have been designed (for example [1, 11, 39, 72, 75, 76]).

However, OT protocols suffer two major drawbacks. The first one is the restriction in the availability of the secret messages, because if the party $\mathcal{B}$ is unavailable, the party $\mathcal{A}$ cannot execute the protocol. This is not the case with DOT protocols which are k-threshold protocols; parts of the secrets are transmitted to m servers, and the receiver may contact any set of t servers ($k \leq t \leq m$) to obtain a secret.

The second drawback of OT protocols is that they cannot be unconditionally secure. Basically, if the receiver's request corresponds to one secret only, the receiver's choice is obviously known by the sender. On the other hand, if the request corresponds to more than one secret, all these secrets need to be transmitted to the receiver, and the sender's security is compromised.

Link with Distributed Oblivious Transfer

It is easy to show that DOT protocols can be reduced to OT protocols. Let π_{OT} be an OT protocol allowing a party $\mathcal{A}$ to obtain one of the n secrets of a party $\mathcal{B}$. The secrets are denoted $\omega_1, \omega_2, \ldots, \omega_n$ and belong to a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power, $p > n$). The secret that $\mathcal{A}$ wishes to obtain is denoted by ω_σ. With each of the m servers involved in π_{OT} is associated an identifier S_j ($j \in \mathcal{I}_{\mathrm{m}} \subseteq \mathbb{K}^*$, $|\mathcal{I}_{\mathrm{m}}| = m$, $1 < m < p$). From each secret ω_i ($1 \leq i \leq n$), $\mathcal{B}$ generates m shares $[\omega_i]_j$ ($j \in \mathcal{I}_{\mathrm{m}}$) thanks to Shamir's secret sharing scheme [89] (see Sect. 2.3.1). Then, $\mathcal{B}$ distributes n shares $[\omega_1]_j, [\omega_2]_j, \ldots, [\omega_n]_j$ to the server S_j ($j \in \mathcal{I}_{\mathrm{m}}$). Now, the situation is that each server S_j holds n secret values $[\omega_i]_j$ ($1 \leq i \leq n$). The party $\mathcal{A}$ selects a set of k server indices $\mathcal{I}_{\mathrm{k}} \subseteq \mathcal{I}_{\mathrm{m}}$ and for each server S_ℓ such that $\ell \in \mathcal{I}_{\mathrm{k}}$, executes π_{OT} to obtain $[\omega_\sigma]_\ell$. From the k collected shares, $\mathcal{A}$ is able to determine the secret ω_σ.

The k instances of π_{OT} can be executed in parallel and the security of the DOT protocol is inherited from the security of the OT protocol. In addition, the DOT protocol's security is guaranteed against coalitions of $k - 1$ servers, thanks to the use of Shamir's secret sharing scheme.

1.2.2 Private Information Retrieval

A *private information retrieval* (PIR) protocol enables a party $\mathcal{A}$ to retrieve one of the data items contained in a database held by a server S, in such a way that S does not learn any information on party $\mathcal{A}$'s request. The first PIR protocol was introduced in 1995 by Chor, Goldreich, Kushilevitz, and Sudan [31] in the information-theoretic setting. Actually, a trivial design consists in sending the whole database to $\mathcal{A}$, which in terms of communication is inefficient. Therefore, research on PIR protocols has focused on minimizing the communications between the party $\mathcal{A}$ and the server S (e.g., [2, 9, 10, 63, 64]). Chor et al. [32] have proved that in an information-theoretic setting, if the database is unique, the trivial solution is optimal. However, if the database is duplicated amongst several servers, assuming these servers do not communicate with each other, and the party $\mathcal{A}$ is allowed to communicate with the servers, then the communication complexity may be sub-linear in n, where n is the size of the database.

It is observed that, unlike DOT protocols (*a*) PIR protocols do not prevent the party $\mathcal{A}$ from obtaining, in addition to the requested item, other items of the database; and (*b*) the servers involved in the protocol are trusted, to the point that they may receive the whole database, not encrypted. A fortiori, coalitions of corrupted servers are not considered.

Some variants of PIR protocols remedy these two shortcomings: the *symmetrically private information retrieval* (SPIR) protocols and the PIR protocols based on the random server model. In a SPIR protocol [55, 56], the party $\mathcal{A}$ does not obtain any information on the data items not chosen; in a PIR protocol based on the random server model, the servers do not receive the whole database, but unintelligible parts of the database [54].

Link with Distributed Oblivious Transfer

It is clear that SPIR protocols and 1-out-of-n OT protocols are equivalent. Since DOT protocols can be reduced to OT protocols, it can be inferred that DOT protocols can be reduced to SPIR protocols. Gertner et al. [55] have proposed both general reductions and specific reductions from a SPIR protocol to a PIR protocol, which implies that DOT protocols may be reduced to PIR protocols. Indeed, Beimel, Chee, Yeow, Wang, and Zhang [8] have recently proposed some efficient reductions of DOT protocols to PIR protocols, maintaining the sub-linear property of underlying PIR protocols.

1.2.3 Multi-party Computation

Secure *multi-party computation* (MPC) allows mutually distrustful parties to jointly compute a function of their inputs, in such a way that the correctness of the output and the privacy of the parties' inputs are guaranteed. The original concept was introduced by

Yao [100], who posed the problem of two millionaires wishing to compute which one is richer, without revealing their wealth. The two-party computation protocol proposed by Yao was later generalized to n parties by Goldreich, Micali, and Wigderson [60].

Thus, if $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ are n parties respectively holding the private inputs x_1, $x_2, \ldots, x_n$ selected in a set S and

$$\begin{aligned} f\colon S^n &\longrightarrow S^n \\ [\, x_1, x_2, \ldots, x_n \,] &\longmapsto [\, y_1, y_2, \ldots, y_n \,] \end{aligned}$$

is a public function, then at the end of the execution of an MPC protocol, the party $\mathcal{P}_i$ ($i = 1, 2, \ldots, n$) obtains the output y_i and has information neither on x_j nor on y_j where $j = 1, 2, \ldots, n$ and $j \neq i$. A great deal of work has been done in this research field, more particularly in the unconditionally secure setting (e.g., [14, 27]). Also, a large body of research has shown that secure MPC is feasible for any computable function (e.g., [14, 27, 60, 99]). Lots of improvements have been achieved since; however, the general protocols tend to be inefficient.

Link with Distributed Oblivious Transfer

Let π_{DOT} be a DOT protocol allowing a receiver to obtain a secret ω_σ from a sender who holds n secrets $\omega_1, \omega_2, \ldots, \omega_n$ selected in a set S. This DOT protocol may be seen as an MPC protocol where the secret input of the sender is $\boldsymbol{u} = [\, \omega_1, \omega_2, \ldots, \omega_n \,]$, the secret input of the receiver is $\boldsymbol{v} = [\, \delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma \,]$ and the function f is defined by

$$\begin{aligned} f\colon \mathbb{K}^n \times \mathbb{K}^n &\longrightarrow \mathbb{K} \times \mathbb{K} \\ [\, \boldsymbol{u}, \boldsymbol{v} \,] &\longmapsto [\, 0, \boldsymbol{u} \cdot \boldsymbol{v} \,] \,. \end{aligned}$$

After the MPC protocol has been executed, the sender has obtained the constant value 0 (i.e., a meaningless value) and the receiver has obtained the value $\boldsymbol{u} \cdot \boldsymbol{v} = \omega_\sigma$, which is exactly the purpose of the protocol π_{DOT}.

1.3 Literature Survey

Currently, there are two main categories of unconditionally secure DOT protocols: the PIR-based DOT protocols and the polynomial interpolation-based DOT protocols. Around 20 protocols were designed and described in only five articles [8, 19, 57, 74, 78] — if extended and improved versions are excluded — between 1997 and 2012. An overview of each of these protocols is given in this section.

1.3.1 Private Information Retrieval-based Distributed Oblivious Transfer Protocols

The first two DOT protocols were introduced by Gertner and Malkin [57] in 1997. Both protocols allow a receiver to obtain one of the n secret bits held by a sender.

The key idea of the first protocol is that the receiver executes a PIR protocol (see Sect. 1.2.2) with k servers in a first round, to collect two pointers. Then, in a second round, the receiver sends one pointer to a first additional server and the second pointer to a second additional server. These two servers use the received pointers to select one bit in a random string of bits. Then, they return the retrieved bits to the receiver. The chosen bit is determined thanks to the 'exclusive or' of the two bits received in the second round. None of the $k + 2$ servers has any information on the receiver's choice, but if two servers collide, they may determine the receiver's choice.

The second protocol is a generalization of the first protocol and aims to guarantee the receiver's privacy against any coalition of t servers or fewer. The protocol requires the receiver to communicate with $kt + t + 1$ servers whereas in the first protocol, the receiver needs to communicate with $k + 2$ servers only. Note that none of the protocols guarantees the secrets' confidentiality against curious or corrupted servers since k servers, in both protocols, receive the sender's secret bits.

The first protocol was improved by Gertner, Ishai, Kushilevitz, and Malkin in 1998 and 2000 [55, 56], so that only one additional server is required instead of two. In addition, Gertner et al. introduced the model of SPIR (see Sect. 1.2.2 above), where besides the privacy of the receiver, the security of the database is also guaranteed against the receiver. More generally, Gertner et al. introduced a general technique to reduce any k-database PIR protocol to a $(k + 1)$-database SPIR protocol, preserving in particular the number of rounds and adding only a constant factor overhead in the communication complexity. They also presented specific reductions, both in honest and malicious settings. However, again, no protection was in place to prevent dishonest coalitions of servers to determine the sender's data. An extension was proposed to protect the sender's privacy against a coalition of t servers, but at the expense of a greater number of servers ($kt + 1$ instead of k servers) or of communication overhead (multiplicative factor of $\binom{k+t-1}{t-1}$ for $k + t$ servers).

The PIR-based technique was abandoned for a few years until 2012, when Beimel, Chee, Wang, and Zhang [8] introduced communication-efficient DOT protocols relying on PIR protocols. Beimel et al. proposed two constructions:

1. The first one is a specific reduction from a polynomial interpolation-based DOT protocol to a polynomial interpolation-based PIR. This construction is presented in the next section.

2. The second construction is a general reduction from a DOT protocol to a PIR protocol. The communication efficiency of the resulting DOT protocol is the same as the efficiency of the underlying PIR protocol. Thus the latest progress in the research of locally decodable codes-based PIR protocols gives efficient sub-linear DOT protocols. For example, if the reduction is applied to the PIR protocol defined by Barkol, Ishai, and Weinreb [3, 4] and Efremenko [46], the resulting DOT protocol's communication complexity is $\exp(O(\sqrt{\log(n\log(\log(n))))}))$ (the receiver must contact $3^{k_{\mathcal{R}}} + k_{\mathcal{R}} + k_{\mathcal{S}}$ servers, where $k_{\mathcal{R}}$ is such that a coalition of up to $k_{\mathcal{R}}$ servers learns no information on the receiver's choice and $k_{\mathcal{S}}$ is such that a coalition of a malicious receiver and $k_{\mathcal{S}}$ malicious servers only obtains one secret after the execution of the protocol). However, in the set-up phase where m servers are involved, the sender has to prepare specific shares for each subset of k servers, giving for some settings of the protocol's parameters an impractical number of elements to transmit to the servers.

1.3.2 Polynomial Interpolation-based Distributed Oblivious Transfer Protocols

In 2000, Naor and Pinkas [74] proposed two unconditionally secure DOT protocols, based on polynomial interpolation. These protocols take non-fully trusted servers into account: servers are only provided with parts — called *shares* — of the original messages. Both protocols are composed of two phases: (*i*) the *set-up phase* and (*ii*) the *transfer phase*. During the set-up phase, the sender generates and sends shares of his secrets to all the servers. In the transfer phase, the receiver chooses the index of a secret, selects k servers ($k > 1$) and sends them requests. From the k responses, the receiver is able to interpolate an univariate polynomial of degree at most $k - 1$, whose value for $x = 0$ is the chosen secret.

In the first protocol, the shares distributed to the servers by the sender are generated from a sparse bivariate polynomial. The receiver's privacy is guaranteed against a coalition of $k - 1$ servers, and with an additional protecting technique, the protocol is secure against a dishonest receiver.

The second protocol proposes a trade-off between the sender's security and the receiver's privacy; the receiver's privacy is guaranteed against a coalition of $k_{\mathcal{R}}$ servers ($0 < k_{\mathcal{R}} < \frac{k-1}{2}$), whereas the sender's security is guaranteed against a coalition between the receiver and $k_{\mathcal{S}} = \frac{k-1}{2k_{\mathcal{R}}}$ corrupt servers. (It is assumed that the design of the protocol takes into consideration the modifications suggested by Ghodosi [58, 59].)

In SAC 2002, Blundo, D'Arco, De Santis, and Stinson presented a generalization of Naor and Pinkas' protocols to n secrets [19]. In 2007, the protocols were improved [20].

Blundo et al. also defined a security model composed of four fundamental conditions that every DOT protocol should satisfy:

B_1. Correctness — The receiver is able to determine the chosen secret once she has received information from the k contacted servers.

B_2. Receiver's privacy — A coalition of up to $k-1$ servers cannot obtain any information on the choice of the receiver.

B_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.

B_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.

The first DOT protocol of Naor and Pinkas [74], as well as the first DOT protocol of Blundo et al. [20], only satisfies security conditions B_1, B_2, and B_3. Actually, Blundo et al. have proved that condition B_4 cannot be guaranteed with a one-round DOT protocol — a round being defined as a set of consistent requests/responses exchanged between the receiver and k servers.

Still in 2002, Nikov, Nikova, Preneel, and Vanderwalle [78] introduced two unconditionally secure DOT protocols based on polynomial interpolation, similar to Blundo et al.'s protocols. The main difference is the sharing polynomial used by the sender to prepare the shares sent to the servers. More generally, Nikov et al. demonstrated that if the receiver's privacy is guaranteed against a coalition of $k_{\mathcal{R}}$ servers and the sender's security against a coalition of $k_{\mathcal{S}}$ servers, including when a secret had already been obtained, then the parameters $k_{\mathcal{S}}$ and $k_{\mathcal{R}}$ must satisfy the inequality $(k_{\mathcal{S}}+1)+(k_{\mathcal{R}}+1)<k$.

In 2007, a new polynomial interpolation-based DOT protocol was introduced by Cheong, Koshiba, and Yoshiyama [29], but it is observed that this protocol is actually an application of the technique suggested by Naor and Pinkas to Nikov et al.'s DOT protocol.

In 2012, Beimel et al. [8] presented a reduction from polynomial interpolation-based DOT protocols to polynomial interpolation-based PIR protocols (for example [9, 98]). The communication complexity of the resulting DOT protocols is sub-linear ($O(n^{1/\lfloor (k-k_{\mathcal{S}}-1)/k_{\mathcal{R}} \rfloor})$ where $k_{\mathcal{R}}$ and $k_{\mathcal{S}}$ are defined as in the previous section), and assumes an 'honest but curious' receiver (see Sect. 2.2.2).

1.3.3 Distributed Oblivious Transfer Protocols Based on Other Techniques

The two polynomial interpolation-based protocols introduced by Blundo et al. [20] were accompanied by three other DOT protocols, based on combinatorial solutions. In the first one, the receiver must contact all servers; the secret bits are hidden in a matrix and each server receives a row of the matrix. The second combinatorial DOT protocol is based on orthogonal arrays and assumes an honest receiver. The third protocol is similar to the second, but is extended to general access structures (authorised lists). Blundo et al. have also described a two-round protocol which satisfies all the security conditions B_1, B_2, B_3, and B_4. The receiver sends a first request to k servers, receives k responses and prepares a second set of requests, based on the received responses. However, this DOT protocol does not guarantee the receiver's privacy, if the sender collides with one server only. Strangely, this scenario is not taken into account by security condition B_2 where the sender does not appear (the condition was generalised by Cheong et al. [29] to take into consideration a coalition of the sender and of servers).

1.4 Motivation and Objectives

As pointed out by Kilian [69], OT is a fundamental cryptographic primitive; this primitive is used as a building block in many secure protocols (e.g., [28, 38, 60, 62]). If DOT protocols enjoy the same properties as OT protocols, they have two advantages over OT protocols: first, they can be designed to be unconditionally secure, provided coalitions of less than k servers are considered, and second, due to their distributed settings, they can easily form the basis of other distributed secure computation protocols.

However, a preliminary analysis shows that most unconditionally secure DOT protocols have weaknesses, flaws or are inefficient, especially in some specific scenarios. Furthermore, when the research on DOT protocols is considered, as well as on a similar cryptographic primitive, namely Secret Sharing [7], the abundant literature on the latter compared to the lack of publications on the former is surprising. Yet, both types of primitives could be designed for the same particular contexts, and so should motivate the design of similar variants. This has been the guiding line of the research whose results are presented in this thesis.

Due to the high number of possible variants of DOT protocols, the research could not be exhaustive and systematic. The research methodology consisted in detecting scenarios of interest, for which a DOT protocol variant would be a contribution to the state of the art, and then to design the corresponding DOT protocol. It is important to note the framework and the limitations of the research:

- Only unconditionally secure DOT protocols were studied, to the detriment of computationally secure DOT protocols. In fact, computationally secure DOT protocols can be easily derived from OT protocols (see Sect. 1.2.1). This is why there is very little research on computationally secure DOT protocols (only one DOT protocol designed by Zhong and Richard Yang [101, 102]).

- The DOT protocols presented in this thesis were all designed in conventional — or traditional — cryptography, rather than in quantum cryptography. Basically, quantum cryptography relies on the physics of the communication carriers [16, 88], while traditional cryptography is based on mathematics. In particular, quantum cryptography protocols are dependent on the medium of communication, which is not the case for traditional cryptography.

- In some security models, correctness is statistical. That is, the receiver obtains the chosen secret with a high probability, $p = 1 - \epsilon$, where $\epsilon = 2^{-\ell}$ and ℓ is an adjustable parameter of the protocol. In the study, it was chosen to define the correctness for a probability of obtaining a secret of $p = 1$ (i.e., $\epsilon = 0$).

- Some cryptographic protocols rely on specific communication physical means or techniques (noisy channel, time-delay, packet reordering, etc.); These particular features were excluded from the study and only conventional communications were considered.

1.5 Contribution and Organization of the Thesis

The first objective of this research was to analyse the principal unconditionally secure DOT protocols. In this thesis, it is shown that some of them are flawed or suffer weaknesses. The second objective was to design DOT variants more efficient or more secure than the existing DOT protocols. The results form the major part of this document and mainly consist of five novel variants of unconditionally secure DOT protocols, suitable to particular environments:

- A DOT protocol with the highest level of security, i.e., secure against an adversary controlling up to $k - 1$ servers. This protocol demonstrates that unconditionally secure, one-round, polynomial interpolation-based DOT protocols, with a receiver corrupting up to $k - 1$ servers once the protocol has been executed, may be secure.

- A verifiable DOT protocol, allowing the receiver to obtain a chosen secret, even in the presence of malicious servers. These servers may try to disrupt the protocol's

execution, either by not responding or by providing incorrect responses, so that the secret obtained by the receiver is different from the chosen secret.

- A t-out-of-n DOT protocol, allowing the receiver to obtain t secrets in one round only. The current one-round DOT protocols allow the receiver to obtain one secret only, but to execute the protocols t times, which is inefficient.

- Two adaptive DOT protocols, where the set-up phase is executed once only, whatever the number of secrets requested by the receiver(s). In particular, these variants are useful when a receiver selects a secret based upon the value of a secret previously obtained. The first variant accepts a limited number of requests, unlike the second one which, however, requires communication amongst servers.

- A threshold DOT protocol, enabling a receiver to contact k servers or more to obtain a secret. The existing DOT protocols require a mechanism to control that exactly k servers are contacted by the receiver, which annihilate the availability feature of DOT protocols compared to OT protocols (see Sect. 1.2.1).

The remainder of this thesis is organized as follows. Chapter 2 introduces the basic concepts common to all DOT protocols presented throughout the document. First, the mathematics (algebra, probability, and information theory) used in the protocols and in the security demonstrations are presented. Second, the models (communication, adversary and security) associated with the protocols are introduced. Third, a few components common to several of the protocols are described. In Chap. 3, some existing DOT protocols are analysed and some weaknesses are pointed out. In the next chapters (Chaps. 4–8), novel variants of protocols are described; each of these five chapters may be read independently of the others. Redundancies have been kept to a minimum, but some were still required to provide consistent and independent chapters. Lastly, conclusions and direction for future research are given in Chap. 9.

When referred to, an appendix may be consulted for detailed proofs, examples or explanations, but is not necessary for the comprehension of the body of the document.

2

Preliminaries

The objective of this chapter is threefold. First, it reviews the mathematical concepts (algebra, probability, and information theory) needed in this thesis. The purpose is to introduce the notation, as well as specificities used in later chapters, but not to give comprehensive treatments of the topics. More detailed information can be found, for example, in the books of Bourbaki [21, 22], Feller [48, 49], Billingsley [17], and Cover and Thomas [33]. Second, the chapter introduces the generic models applied in the rest of the document, that is the communication model, the security model and the adversary model. Lastly, the basic cryptographic components, common to some of the protocols presented in this thesis, are described. These components are Shamir's secret sharing scheme [89], as well as a simplified version and the full version of the polynomial interpolation-based distributed oblivious transfer protocol introduced by Blundo, D'Arco, De Santis, and Stinson [19, 20].

2.1 Background Mathematics

2.1.1 Algebra

2.1.1.1 Modular Arithmetic

A fundamental characteristic of many cryptographic protocols is that one party holds secret information. It could be, for example, secret messages, secret words, secret numbers or secret bits. In modern cryptographic systems, secrets are often represented as numerical values, and more specifically, integers. These mathematical entities are easy to manipulate with the usual arithmetic operations (addition, subtraction, multiplication and division).

Moreover, simple relations like $A \leftrightarrow 1$, $B \leftrightarrow 2$, ..., $Z \leftrightarrow 26$, $a \leftrightarrow 27$, $b \leftrightarrow 28$, ..., $z \leftrightarrow 52$, " " (space) $\leftrightarrow 53$, "." $\leftrightarrow 54$, "," $\leftrightarrow 55$, etc. allow the conversion of numbers into texts and vice versa.

Some of the protocols presented in this thesis rely on Lagrange interpolation. Therefore, the secrets are required to belong to a commutative field, so that every non-zero element has a multiplicative inverse. Basically, a field $\mathbb{K}$ is a set of elements with two laws of composition (an addition, commutative and associative, and a multiplication, associative). The field is commutative if the multiplication is commutative. One of the elements of $\mathbb{K}$ is the additive identity (denoted by 0, such that $x = x + 0 = 0 + x$ for any element x of $\mathbb{K}$) and another one is the multiplicative identity (denoted by 1, such that $x = x \times 1 = 1 \times x$ for any element x of $\mathbb{K}$). Moreover, each element of $\mathbb{K}$ has an additive inverse (i.e., if x is an element of $\mathbb{K}$, there exists an element y such that $x + y = y + x = 0$) and each element of $\mathbb{K}^*$ has a multiplicative inverse (i.e., if x is an element of $\mathbb{K}$, different from 0, there exists an element y such that $x \times y = y \times x = 1$). Lastly, the multiplication must be distributive over the addition, that is, for any 3 elements x, y and z of $\mathbb{K}$, $x \times (y + z) = x \times y + x \times z$.

Besides, it is sometimes necessary to select a random number in the set of values containing the secrets of the protocols. Indeed, it is impossible to choose a random number in an infinite set of numbers in a way that all numbers are equally likely (the sum of the probabilities cannot be 1). Hence, secrets will belong to finite sets of numbers.

From the two requirements above, it follows that secrets are selected in finite fields, i.e., Galois fields $\mathrm{GF}(q^n)$ where q is prime and n a positive integer (the modulus q^n of the field is called a prime power). The immediate consequence is that modular arithmetic will be used, and not real arithmetic. To simplify notations, congruence symbols will be omitted in operations (e.g., if $q = 7$ and $n = 1$, instead of '$3 + 5 \equiv 1 \pmod 7$' it will be written '$3 + 5 = 1$', keeping in mind that the modulus of the field is $q^n = 7$).

Throughout this document, all operations on secrets are executed in a finite field $\mathbb{K} = [\,\mathrm{GF}(p), +, \times\,]$, where $p \in \mathbb{N}$ is a prime power, $+$ is the additive law of composition, and $\times$ is the multiplicative law of composition of the field. When p is prime, it is assumed that $\mathrm{GF}(p) = \mathbb{Z}/p\mathbb{Z} = \{\,\overline{0}, \overline{1}, \ldots, \overline{p-1}\,\}$ where the elements of $\mathbb{Z}/p\mathbb{Z}$ are representatives of the equivalence classes for the congruence modulo p relation. More precisely, an element $\overline{e}$ belongs to $\mathbb{Z}/p\mathbb{Z}$ if e is the smallest non-negative integer of its class. In this context, it is convenient to define an order relation '$\leq$' on the set $\mathrm{GF}(p) = \mathbb{Z}/p\mathbb{Z}$: for $\overline{e_1} \in \mathbb{Z}/p\mathbb{Z}$ and $\overline{e_2} \in \mathbb{Z}/p\mathbb{Z}$, $\overline{e_1} \leq \overline{e_2}$ if $e_1 \leq e_2$.

2.1.1.2 Polynomials

Some underlying mechanisms of the protocols described in this document are based on polynomial interpolation. A polynomial with coefficients in a group $\mathbb{G}$ may be considered

as an infinite sequence of elements of $\mathbb{G}$, with a finite number of non-zero elements. It is convenient to introduce the indeterminate X and to represent a polynomial, not as a sequence, but as a sum of products between a coefficient of $\mathbb{K}$ and a power of the indeterminate. From this point, this notation will be used. For example, if $\mathbb{G} = \mathbb{Z}$, the polynomial $\boldsymbol{F} = [\,1, 5, 7, 0, 3, 0, 0, \ldots\,]$ will be written under the form $F = 1 + 5X + 7X^2 + 3X^4$.

Definition 2.1. *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* $\mathbb{K}$*, a polynomial* $F = \sum_{i=0}^{k} f_i X^i$ *of* $\mathbb{K}_k[X]$ *is said to be* 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.*

Definition 2.2. *By an abuse of language, if* $F = \sum_{i=0}^{r} f_i X^i$ $(r \in \mathbb{N}, f_i \in \mathbb{K})$ *is a polynomial of* $\mathbb{K}[X]$*, the polynomial function* F *is given by* $F : \mathbb{K} \to \mathbb{K}$, $x \mapsto \sum_{i=0}^{r} f_i x^i$*. Note that the additive and multiplicative laws of composition in* $\mathbb{K}[X]$ *are defined such that, for* P, Q, $R \in \mathbb{K}[X]$ *and for* $x \in \mathbb{K}$

- *If* $R = P + Q$*, then the corresponding polynomial functions satisfy the relationship* $R(x) = P(x) + Q(x)$*,*
- *Similarly, if* $R = P \times Q$*, then the corresponding polynomial functions satisfy the relationship* $R(x) = P(x) \times Q(x)$*.*

2.1.1.3 Lagrange's Interpolation Technique

Let $\mathbb{K}$ be a field and $\mathbb{K}[X]$ be the ring of polynomials with coefficients in $\mathbb{K}$. According to Lagrange's interpolation theorem, given k points $[\,a_1, b_1\,], [\,a_2, b_2\,], \ldots, [\,a_k, b_k\,]$ of $\mathbb{K}^2$, where abscissae a_i are distinct, a polynomial R_G agreeing with the k points has the form

$$R_G = \sum_{i=1}^{k} b_i L_i + G \times \prod_{j=1}^{k} (X - a_j)\,,$$

where G can be any polynomial of $\mathbb{K}[X]$ and where the polynomials L_i are defined by:

$$L_i = \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{X - a_j}{a_i - a_j}\,.$$

Note that $L_i(a_j) = \delta_i^j$, $L_i \in \mathbb{K}_{k-1}[X]$ and $R_0 = \sum_{i=1}^{k} b_i L_i$ is the unique polynomial of $\mathbb{K}_{k-1}[X]$ agreeing with the points.

Moreover, to determine the value of the interpolating polynomial $R_0 \in \mathbb{K}_{k-1}[X]$ for the value $x = 0$, it is more efficient to calculate

$$R_0(0) = \sum_{i=1}^{k} b_i L_i(0)\,,$$

where

$$L_i(0) = \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{-a_j}{a_i - a_j}.$$

2.1.2 Probability Theory

2.1.2.1 Finite Probability Space

In Sect. 2.1.1.1, it was mentioned that the secret elements of cryptographic protocols are selected in finite fields. For the sake of generality, it is assumed that an element of a finite field may be chosen according to any probability distribution. This naturally leads to the use of discrete probability theory concepts, applied to finite sets of possible secrets. These concepts are briefly described below.

A *finite probability space* $[\,\Omega, \Sigma, \Pr\,]$ consists of a finite set $\Omega \neq \varnothing$ (the *sample space*), a collection Σ of subsets of Ω (the *event space*), together with a function Pr (the *probability measure*). The elements of Ω are called *elementary events* and the elements of Σ are called *events*.

The event space Σ is closed under complement operations and countable unions, that is

1. $\Sigma \neq \varnothing$
2. If $A \in \Sigma$ then $\Omega \setminus A \in \Sigma$
3. If $A_1, A_2, \ldots, A_n$ are n events of Σ ($n \geq 1$) then

$$\left(\bigcup_{i=1}^{n} A_i \right) \in \Sigma$$

The probability measure Pr is a function whose domain is Σ and image is $\mathbb{R}$, and which satisfies the Kolmogorov axioms:

1. $\Pr(A) \geq 0$ for any event $A \in \Sigma$
2. $\Pr(\Omega) = 1$
3. If A and B are two events of Σ such that $A \cap B = \varnothing$,

$$\Pr(A \cup B) = \Pr(A) + \Pr(B)$$

It is easy to deduce from these axioms the following two properties: $\Pr(\varnothing) = 0$ and $0 \leq \Pr(A) \leq 1$ for any $A \in \Sigma$. Commonly, the real number $\Pr(A)$ is called the *probability* of the event A. The function Pr also satisfies the essential property,

$$\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B),$$

where A and B are two events of Σ.

Mutual Independence. Two events A and B of Σ are *independent* if

$$\Pr(A \cap B) = \Pr(A) \times \Pr(B)\,.$$

Conditional Probability. The *conditional probability* $\Pr(A \mid B)$ of an event A given that another event B occurs is defined as

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)}$$

whenever $\Pr(B)$ is positive. In the rest of the document, the standard notation $\Pr(A, B) = \Pr(A \cap B)$ will be used.

2.1.2.2 Discrete Random Variables

A *discrete random variable* $\boldsymbol{X}$ on a discrete probability space $[\,\Omega, \Sigma, \Pr\,]$ is a function

$$\boldsymbol{X} : \Omega \longrightarrow \mathcal{X}$$

such that $\{\,\omega \in \Omega \mid \boldsymbol{X}(\omega) = x\,\} \in \Sigma$ for $x \in \mathcal{X}$. It is assumed that the domain $\boldsymbol{X}(\Omega)$ is such that $\mathcal{X} = \boldsymbol{X}(\Omega)$ and the alphabet of $\boldsymbol{X}$ is denoted by $\mathcal{X}$. For convenience, the event of Σ,

$$\{\,\omega \in \Omega \mid \boldsymbol{X}(\omega) = x\,\},$$

is denoted by $\boldsymbol{X} = x$. Hence, the *probability mass function* $p_{\boldsymbol{X}}$ of the discrete random variable $\boldsymbol{X}$ can be defined:

$$\begin{aligned} p_{\boldsymbol{X}} : \mathcal{X} &\longrightarrow \mathbb{R}^+ \\ x &\longmapsto \Pr(\boldsymbol{X} = x)\,. \end{aligned}$$

Actually, the use of random discrete variables allows the omission of theoretical details of probability measures and the use of probability mass functions defined on alphabets. Thus, in the rest of the thesis, when a random variable $\boldsymbol{X}$ (bold font upper case letter) is introduced, it will implicitly be associated with an alphabet $\mathcal{X}$ (cursive font)

and a probability mass function $p_{\boldsymbol{X}}$ (lower case p with, as subscript, the random variable identifier), while the other components, e.g., the sample space Ω, the event space Σ and even the probability measure Pr, will be ignored. Because the most important feature of a discrete random variable is its probability mass function, a few properties of random variables are introduced below. These properties are directly induced from equivalent properties of probability measures (see previous section).

Joint Probability. If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are two discrete random variables defined on the same probability space, the pair $[\,\boldsymbol{X}, \boldsymbol{Y}\,]$ can be associated with a *joint probability mass function*. This function, denoted by $p_{\boldsymbol{XY}}$, is defined by

$$p_{\boldsymbol{XY}}(x, y) = \Pr(\boldsymbol{X} = x, \boldsymbol{Y} = y)\,.$$

for any pair $[\,x, y\,]$ on $\mathcal{X} \times \mathcal{Y}$. Note that this definition can be extended to any tuple of discrete random variables.

Given a probability mass function p_{XY} for $\boldsymbol{X}$, $\boldsymbol{Y}$, one can recover the marginal distribution of $\boldsymbol{X}$ via

$$p_{\boldsymbol{X}}(x) = \sum_{y \in \mathcal{Y}} p_{\boldsymbol{XY}}(x, y)\,.$$

This formula can be generalized to any finite tuple of random variables. Thus, let $\boldsymbol{X_i}$ be a random variable over an alphabet $\mathcal{X}_i$, with a probability mass function $p_{\boldsymbol{X_i}}$ $(0 \leq i \leq k)$. Similarly, let $\boldsymbol{Y_j}$ be a random variable over an alphabet $\mathcal{Y}_j$, with a probability mass function $p_{\boldsymbol{Y_j}}$ $(0 \leq j \leq \ell)$. If $p_{\boldsymbol{X_1 X_2 \ldots X_k Y_1 Y_2 \ldots Y_\ell}}$ is the joint probability mass function between all these variables, then the marginal probability mass function $p_{\boldsymbol{X_1 X_2 \ldots X_k}}$ is defined and calculated thanks to the formula:

$$p_{\boldsymbol{X_1 X_2 \ldots X_k}}(x_1, x_2, \ldots, x_k) = \sum_{\substack{y_1 \in \mathcal{Y}_1, \\ y_2 \in \mathcal{Y}_2 \\ \ldots \\ y_\ell \in \mathcal{Y}_\ell}} p_{\boldsymbol{X_1 X_2 \ldots X_k Y_1 Y_2 \ldots Y_\ell}}(x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_\ell) \tag{2.1}$$

where $x_i \in \mathcal{X}_i$ for $i = 1, 2, \ldots, k$.

From this point, all random variables introduced in the document will be implicitly discrete.

Conditional Probability. For two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ associated with a joint probability mass function $p_{\boldsymbol{XY}}$, the conditional probability $p_{\boldsymbol{X|Y}}$ of the random variable $\boldsymbol{X}$ given the random variable $\boldsymbol{Y}$ is defined as

$$p_{\boldsymbol{X|Y}}(x \mid y) = \Pr(\boldsymbol{X} = x \mid \boldsymbol{Y} = y),$$

for any pair $[\, x, y \,] \in \mathcal{X} \times \mathcal{Y}$, such that $p_{\boldsymbol{Y}}(y) \neq 0$, i.e.,

$$p_{\boldsymbol{X}|\boldsymbol{Y}}(x \mid y) = \frac{p_{\boldsymbol{XY}}(x, y)}{p_{\boldsymbol{Y}}(y)} \,. \tag{2.2}$$

Thus, for each value $y \in \mathcal{Y}$ such that $p_{\boldsymbol{X}}(y) \neq 0$, there is a probability mass function of $\boldsymbol{X}$ given that $\boldsymbol{Y} = y$, denoted by

$$p_{\boldsymbol{X}|\boldsymbol{Y}=y}(x) = p_{\boldsymbol{X}|\boldsymbol{Y}}(x \mid y) = \Pr(\boldsymbol{X} = x \mid \boldsymbol{Y} = y) \,.$$

Mutual Independence. Two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ are said to be independent if for all pairs $[\, x, y \,] \in \mathcal{X} \times \mathcal{Y}$,

$$p_{\boldsymbol{XY}}(x, y) = p_{\boldsymbol{X}}(x) \times p_{\boldsymbol{Y}}(y) \,.$$

2.1.3 Theory of Information

Since security conditions are linked to the quantity of information received by parties, it seems appropriate to use Shannon's entropy function [90, 91], and more generally information theory, to demonstrate the security of the protocols I designed. The following definitions and properties will be used in the rest of the document (for more details on information theory, see for example [33]).

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two random variables over an alphabet $\mathcal{V} = \mathbb{K}$, where $\mathbb{K}$ is a finite field, and let $p_{\boldsymbol{XY}}$ be their joint probability mass function. The probability mass functions $p_{\boldsymbol{X}}$ and $p_{\boldsymbol{Y}}$ associated with $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively are determined thanks to (2.1).

- The *entropy* of $\boldsymbol{X}$ is

$$H(\boldsymbol{X}) = -\sum_{x \in \mathcal{X}} p_{\boldsymbol{X}}(x) \log_2 p_{\boldsymbol{X}}(x) \,.$$

- The *joint entropy* $H(\boldsymbol{X}, \boldsymbol{Y})$ of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is

$$H(\boldsymbol{X}, \boldsymbol{Y}) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{\boldsymbol{XY}}(x, y) \log_2 p_{\boldsymbol{XY}}(x, y) \,.$$

- The *conditional entropy* $H(\boldsymbol{X} \mid \boldsymbol{Y})$ of $\boldsymbol{X}$ given $\boldsymbol{Y}$ is defined as

$$H(\boldsymbol{X} \mid \boldsymbol{Y}) = \sum_{y \in \mathcal{Y}} p_{\boldsymbol{Y}}(y) H(\boldsymbol{X} \mid \boldsymbol{Y} = y),$$

  where the entropy $H(\boldsymbol{X} \mid \boldsymbol{Y} = y)$ is

$$H(\boldsymbol{X} \mid \boldsymbol{Y} = y) = -\sum_{x \in \mathcal{X}} p_{\boldsymbol{X}|\boldsymbol{Y}}(x \mid y) \log_2 p_{\boldsymbol{X}|\boldsymbol{Y}}(x \mid y) \,.$$

Note that if $\boldsymbol{R}$ is a random variable and $p_{\boldsymbol{R}}(r) = 0$, then the convention that $p_{\boldsymbol{R}}(r) \log_2 p_{\boldsymbol{R}}(r) = 0$ is adopted.

Let $\boldsymbol{X}, \boldsymbol{Y}$, and $\boldsymbol{Z}$ be random variables. The following properties are used in the security demonstrations:

$$H(\boldsymbol{X}) \geq H(\boldsymbol{X} \mid \boldsymbol{Y}) \tag{2.3}$$

$$H(\boldsymbol{X}, \boldsymbol{Y}) = H(\boldsymbol{X}) + H(\boldsymbol{Y} \mid \boldsymbol{X}) = H(\boldsymbol{Y}) + H(\boldsymbol{X} \mid \boldsymbol{Y}) \tag{2.4}$$

$$0 \leq H(\boldsymbol{X}) \leq \log_2 |\mathcal{X}| \tag{2.5}$$

$$\text{If } H(\boldsymbol{Y} \mid \boldsymbol{Z}) = 0 \text{ then } H(\boldsymbol{X} \mid \boldsymbol{Y}) \geq H(\boldsymbol{X} \mid \boldsymbol{Z}) \tag{2.6}$$

$$\text{If } H(\boldsymbol{Y} \mid \boldsymbol{Z}) = 0 \text{ then } H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = H(\boldsymbol{X} \mid \boldsymbol{Z}) \tag{2.7}$$

$$H(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{Y}) = H(\boldsymbol{Z} \mid \boldsymbol{X}) \text{ iff } H(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{Z}) = H(\boldsymbol{Y} \mid \boldsymbol{X}) \tag{2.8}$$

$$\text{If } H(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{Y}) = H(\boldsymbol{Z}) \text{ then } H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = H(\boldsymbol{X} \mid \boldsymbol{Y}) \tag{2.9}$$

and

$$\text{If } H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = H(\boldsymbol{X}) \text{ then } H(\boldsymbol{X} \mid \boldsymbol{Y}) = H(\boldsymbol{X} \mid \boldsymbol{Z}) = H(\boldsymbol{X}) \tag{2.10}$$

2.2 Models

This section introduces the communication model, adversary model and security model used in the rest of the document. The communication model defines how the different entities of protocols communicates with each other. The adversary model gives a classification of adversaries who would wish to breach the security of protocols. Finally, the security model defines how the security of the protocols is assessed and more importantly, defines the security criteria which are evaluated, as well as the specific parameters associated with each criterion.

2.2.1 Communication Model

It is considered that all entities (sender, receiver, servers, etc.) involved in the protocols introduced in this thesis are connected to the same network allowing them to communicate with each other. For each DOT protocol are made some assumptions on the network characteristics; these characteristics are shortly presented below.

First, the entities may send messages to other entities through *unicast* channels or *broadcast* channels. A unicast or point-to-point channel enables both way communication between two entities. A broadcast or point-to-multipoint channel allows any party to send a message to all other entities in the network.

Second, the topology of the network connecting the protocols' entities is *complete*, i.e., any two entities are connected — in that they can exchange messages. In particular, it is assumed that some mechanisms guarantee the connexity of the network so that a receiver cannot execute two instances of a DOT protocol in two disjoint parts of the network, while she is entitled to execute one instance only.

Third, communication channels may be *secure*, *insecure* or *unauthenticated*. A secure channel guarantees that communications cannot be tampered with and cannot be eavesdropped on. With insecure channels, an adversary can obtain all the messages exchanged by the entities; yet, the adversary cannot alter the communication. Finally, unauthenticated channels allow an adversary to have a full control on communication: in addition to the ability to eavesdrop on messages, the adversary may modify or delete messages, and even generate messages on behalf of an entity. An immediate consequence of these definitions is that broadcast channels are not secure since all entities, including an adversary, receive broadcast messages.

Communication network are also characterized by the synchronisation of messages; Basically, a network may be *synchronous* or *asynchronous*. A network is synchronous if all the entities connected to the network have access to a common clock; time is divided by equidistant clock ticks, and messages are sent on a clock tick and received at the next clock tick. That is, the delay of messages in the channel is bounded by a known constant. In asynchronous networks, there is neither common clock, nor bounded delay for the reception of a message. In particular the messages may be received in an order different from the order of sending. In this thesis, networks are assumed to be asynchronous which is more general than considering synchronous networks [13].

2.2.2 Adversary Model

An adversary represents the coalition formed by a party who tries to breach the security of a cryptographic protocol (i.e., who tries to obtain information that should remain private) and possibly, by some of the parties taking part in the protocol. The parties under the control of the adversary are said to be corrupted. In particular, in a DOT protocol, the adversary may be the sender, the receiver or a party external to the protocol.

Informally, the adversary model specifies a list of assumptions on the adversary's capability.

An adversary may be *passive*, *semi-honest* or '*honest but curious*' if the corrupt parties do not interfere with the execution of the protocol, but are able to collect data (internal states, temporary and final computation results, messages sent and received, etc.) and provide these data to the adversary.

An adversary may also be *active* or *malicious*, that is, the adversary can make the corrupted parties behave arbitrarily, and interfere with the communication by deleting, injecting, or modifying messages.

Regarding the strategy of corruption, an adversary may also be categorized as *static* or *adaptive*. A static adversary cannot corrupt parties during the execution of the protocol. Hence the set of corrupted parties is fixed before the protocol starts. The adversary is adaptive (or dynamic) if the parties can be corrupted during the execution of the protocol. If the number of corrupted parties is limited, at any moment, there cannot be more parties that the limit colluding with the adversary, but the set of corrupt parties may change in the course of the protocol's execution. For all the DOT protocols presented in this document, the adversary model is static.

An adversary is also characterized by the computational resources at his disposal (bounded or unbounded). Because all the protocols presented in this thesis are unconditionally secure, only adversaries with unlimited computational resources are considered.

2.2.3 Security Model

The security of a DOT protocol may be assessed thanks to the following security conditions based on definitions given by Blundo, D'Arco, De Santis, and Stinson [19, 20]:

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

C_2. *Receiver's privacy* – A coalition of up to λ_R servers ($1 \leq \lambda_R \leq m$) 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 ($1 \leq \lambda_S \leq m$) 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 ($0 \leq \lambda_C \leq m$) 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 ($\lambda_C > 0$).

In [20], a DOT protocol is private if the following 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 a one-round DOT protocol cannot reach strong privacy, i.e, if $t = k$ and $\lambda_R = k - 1$, then condition $C_{4.2}$ cannot be satisfied for $\lambda_C = k - 1$. More generally, Nikov, Nikova, Preneel, and Vandewalle [78] have demonstrated that the relation $\lambda_R + \lambda_C < k$ needs to hold for conditions C_2 and $C_{4.2}$ to be guaranteed.

2.3 Some Useful Components

In this section, a few basic cryptographic components are gathered. Some of the protocols described in the rest of the document are based on these components. To avoid repetitions, the components are described only once here. The first component, Shamir's secret sharing scheme [89], is a threshold protocol allowing cooperating parties holding parts of a secret to determine this secret only when a threshold of parties is reached. The second component is the polynomial interpolation-based DOT protocol introduced by Blundo et al. [19], because some of the protocols I designed are simply a modified version of this DOT protocol. A basic version of the protocol (also referenced as *sub-protocol*) is described first, and then the full version of the protocol, which is actually an enhancement of the basic protocol, is presented.

2.3.1 Shamir's Secret Sharing Scheme

In [89], Shamir shows how to share a secret ω randomly selected in a prime finite field amongst m users $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ such that ω can be reconstructed from any set of k shares ($k \leq m$). The construction is called a secret sharing scheme. More specifically, because ω can be determined from any set containing k or more shares, k is called the threshold of the scheme and the scheme itself is called a (k, m)-threshold secret sharing scheme. Shamir suggests the following algorithm for his construction:

1. A dealer, $\mathcal{D}$, who has a secret ω in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime), secretly chooses $k - 1$ random elements of $\mathbb{K}$, denoted by $a_1, a_2, \ldots, a_{k-1}$ and forms the *sharing polynomial* $F = \sum_{\ell=1}^{k-1} a_\ell X^\ell + \omega$.

2. $\mathcal{D}$ gives (in private) the share $F(i)$ to the user $\mathcal{U}_i$.

In the secret reconstruction phase, a set of at least k participants uses Lagrange's interpolation formula (see Sect. 2.1.1.3) to recover the secret. Without loss of generality,

let $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k$ be the set of collaborating users in the secret reconstruction phase. Then the secret can be recovered, using

$$\omega = \sum_{i=1}^{k} F(i) \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{j}{j-i}.$$

It is well-known that Shamir's secret sharing scheme is perfect, i.e., the knowledge of $k-1$ or fewer shares leaves ω completely undetermined (see information-theoretic demonstration in App. A).

Remark 2.3. *In Shamir's scheme, the secret ω is randomly selected in a finite field $GF(p)$ where $p \in \mathbb{N}$ is prime. Actually, the scheme is still valid in any finite field, i.e., in Galois fields where p is a prime power, and for any secret ω whose probability distribution is not uniform.*

2.3.2 Blundo et al.'s Distributed Oblivious Transfer Protocol (Simplified Protocol)

In SAC 2002, Blundo et al. [19] presented an extension of Naor and Pinkas' 1-out-of-2 DOT protocol [74]. This polynomial interpolation-based protocol (see description in Fig. 2.1) allows a receiver to obtain one of the n secrets ($n > 1$) of a sender, by contacting exactly k servers. The secrets belong to a field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime) and the parameter k is such that $1 < k \leq m$, where m is the total number of servers participating to the protocol. In terms of security, without additional features, the protocol only guarantees security conditions C_1 and C_2 for $\lambda_{\mathrm{R}} = k - 1$ in a semi-honest model (security conditions and parameters are defined in Sect. 2.2.3).

2.3.3 Blundo et al.'s Distributed Oblivious Transfer Protocol (Full Protocol)

Blundo et al. later designed a new DOT protocol [20] which improved the security of the protocol described in the previous section. The new protocol is still polynomial interpolation-based, but with only one new assumption (the secrets need to be distinct), it guarantees security conditions C_1, C_2 for $\lambda_{\mathrm{R}} = k - 1$ and C_3 for $\lambda_{\mathrm{S}} = k - 1$. Blundo et al. claimed that the protocol satisfies security condition $C_{4.1}$, but this claim is correct only if the protocol is slightly modified as shown in Chap. 3.

The two phases of the protocol — the set-up phase and the transfer phase — are described in Figs. 2.2 and 2.3.

Let $\mathcal{I}_{\mathrm{m}} = \{\, 1, 2, \ldots, m \,\}$ $(m > 1)$ be a subset of $\mathbb{K}^*$ and S_j be a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Input The sender $\mathcal{S}$ contributes with n secrets $\omega_0, \omega_1, \ldots, \omega_{n-1}$ selected in $\mathbb{K}$
The receiver $\mathcal{R}$ chooses an index $\sigma \in \mathcal{I}_{\mathrm{n}} = [\![\, 0, n-1 \,]\!]$, and contributes with $n-1$ private values $\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma$ of $\{\, 0, 1 \,\}$

Output $\mathcal{R}$ receives ω_σ, while $\mathcal{S}$ receives nothing.

Set-up Phase

1. The sender $\mathcal{S}$ generates a quasi-random polynomial B_0 of $\mathbb{K}_{k-1}[X]$, such that $B_0(0) = \omega_0$, and the sparse n-variate polynomial Q defined by

$$Q(x, y_1, y_2, \ldots, y_{n-1}) = B_0(x) + \sum_{j=1}^{n-1} (\omega_j - \omega_0) \times y_j \,.$$

   Note that $\omega_\ell = Q(0, \delta_1^\ell, \delta_2^\ell, \ldots, \delta_{n-1}^\ell)$ for $\ell \in \mathcal{I}_{\mathrm{n}}$.

2. Then, to each server S_j $(j \in \mathcal{I}_{\mathrm{m}})$, $\mathcal{S}$ transmits the $(n-1)$-variate polynomial F_j defined by

$$F_j(y_1, y_2, \ldots, y_{n-1}) = Q(j, y_1, y_2, \ldots, y_{n-1}) \,.$$

Transfer Phase

1. The receiver $\mathcal{R}$ chooses the identifier σ of one secret $(\sigma \in \mathcal{I}_{\mathrm{n}})$ and generates $n-1$ quasi-random polynomials Z_i $(1 \le i \le n-1)$ of $\mathbb{K}_{k-1}[X]$, such that $Z_i(0) = \delta_i^\sigma$.

2. Then, $\mathcal{R}$ selects a subset $\mathcal{I}_k \subseteq \mathcal{I}_{\mathrm{m}}$ of k indices and sends to each server S_j $(j \in \mathcal{I}_k)$ a request $\boldsymbol{r}_j = [\, Z_1(j), Z_2(j), \ldots, Z_{n-1}(j) \,]$. When a server S_j receives $\boldsymbol{r}_j$, it replies with the share $s_j = F_j(\boldsymbol{r}_j)$.

3. After receiving k responses s_j $(j \in \mathcal{I}_k)$, $\mathcal{R}$ interpolates a univariate polynomial R from the k points $\left[\, j, s_j \,\right]$ (see Sect. 2.1.1.3) and calculates the chosen secret: $\omega_\sigma = R(0)$.

Figure 2.1: Blundo et al.'s Simplified Distributed Oblivious Transfer Protocol

Let $\mathcal{I}_{\mathrm{m}} = \{ 1, 2, \ldots, m \}$ ($m > 1$) be a subset of $\mathbb{K}^*$ and S_j be a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Input The sender $\mathcal{S}$ contributes with:

- n secrets $\omega_0, \omega_1, \ldots, \omega_{n-1}$ of $\mathbb{K}$,
- $n-1$ masks $u_1^1, u_2^1, \ldots, u_{n-1}^1$ of $\mathbb{K}$,
- $n-1$ masks $u_1^2, u_2^2, \ldots, u_{n-1}^2$ of $\mathbb{K}$, and
- n masks $v_0, v_1, \ldots, v_{n-1}$ of $\mathbb{K}^*$.

The receiver $\mathcal{R}$ chooses an index $\sigma \in \mathcal{I}_{\mathrm{n}} = [\![0, n-1]\!]$, and contributes with $n-1$ private values $\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma$ of $\{ 0, 1 \}$

Output $\mathcal{R}$ receives ω_σ, while $\mathcal{S}$ receives nothing.

Set-up Phase

1. The sender $\mathcal{S}$ generates two quasi-random polynomials B_0^1 and B_0^2 of $\mathbb{K}_{k-1}[X]$, such that $B_0^1(0) = v_0\omega_0$ and $B_0^2(0) = v_0$. Then, $\mathcal{S}$ generates two sparse n-variate polynomial Q^1 and Q^2:

$$\begin{cases} Q^1(x, y_1, y_2, \ldots, y_{n-1}) = B_0^1(x) + \displaystyle\sum_{j=1}^{n-1} (u_j^1 v_j \omega_j - v_0\omega_0) \times y_j \,, \\ Q^2(x, y_1, y_2, \ldots, y_{n-1}) = B_0^2(x) + \displaystyle\sum_{j=1}^{n-1} (u_j^2 v_j - v_0) \times y_j \,. \end{cases}$$

   For each mask u_i^ℓ ($\ell = 1, 2$, $i \in [\![1, n-1]\!]$), $\mathcal{S}$ generates m shares $[u_i^\ell]_j$ ($j \in \mathcal{I}_{\mathrm{m}}$) thanks to Shamir's secret sharing scheme (see Sect. 2.3.1).

2. Then, to each server S_j ($j \in \mathcal{I}_{\mathrm{m}}$), $\mathcal{S}$ transmits the $(n-1)$-variate polynomial F_j^i ($i = 1, 2$) defined by

$$F_j^i(y_1, y_2, \ldots, y_{n-1}) = Q^i(j, y_1, y_2, \ldots, y_{n-1}),$$

   as well as the shares $[u_i^\ell]_j$ ($\ell = 1, 2$, $i \in [\![1, n-1]\!]$).

Figure 2.2: Blundo et al.'s Full Distributed Oblivious Transfer Protocol - Set-up Phase

Transfer Phase

1. The receiver $\mathcal{R}$ chooses the identifier σ of one secret ($\sigma \in \mathcal{I}_{\mathrm{n}}$) and generates $n-1$ quasi-random polynomials Z_i ($1 \leq i \leq n-1$) of $\mathbb{K}_{k-1}[X]$, such that $Z_i(0) = \delta_i^\sigma$.
2. Then, $\mathcal{R}$ selects a subset $\mathcal{I}_k \subseteq \mathcal{I}_{\mathrm{m}}$ of k indices and sends to each server S_j ($j \in \mathcal{I}_k$) a request $\boldsymbol{r}_j = [\, Z_1(j), Z_2(j), \ldots, Z_{n-1}(j) \,]$. When a server S_j receives $\boldsymbol{r}_j$, it replies with the two shares $s_j^1 = F_j^1(\boldsymbol{r}_j)$, $s_j^2 = F_j^2(\boldsymbol{r}_j)$, the $n-1$ shares $[u_i^1]_j$ and the $n-1$ shares $[u_i^2]_j$ ($i \in [\![1, n-1]\!]$).
3. After receiving k responses, if $\sigma = 0$, $\mathcal{R}$ applies Lagrange's interpolation technique to determine $R^1(0) = v_0\omega_0$ from the k points $\left[j, s_j^1 \right]$ and $R^2(0) = v_0$ from the k points $\left[j, s_j^2 \right]$. With a simple division, $\mathcal{R}$ obtains ω_0. If $\sigma \neq 0$, $\mathcal{R}$ applies the reconstruction phase of Shamir's secret sharing scheme to determine u_σ^1 and u_σ^2. Then, $\mathcal{R}$ applies Lagrange's interpolation technique (see Sect. 2.1.1.3) to determine $R^1(0) = u_\sigma^1 v_\sigma \omega_\sigma$ from the k points $\left[j, s_j^1 \right]$ and $R^2(0) = u_\sigma^2 v_\sigma$ from the k points $\left[j, s_j^2 \right]$. The secret ω_σ is then obtained thanks to the equality

$$\omega_\sigma = \frac{u_\sigma^2 R^1(0)}{u_\sigma^1 R^2(0)} .$$

Figure 2.3: Blundo et al.'s Full Distributed Oblivious Transfer Protocol - Transfer Phase

3

Security Analysis of Polynomial Interpolation-based Distributed Oblivious Transfer Protocols

In an unconditionally secure *distributed oblivious transfer* 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 distributed oblivious transfer protocols are used in auctions, some bids may be more probable than others.

In this kind of scenario, it is shown 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 distributed oblivious transfer protocols allow a cheating receiver, or a curious server, who has obtained a linear combination of the secrets to determine all the secrets.

3.1 Introduction

In their generalization of Even, Goldreich, and Lempel's *oblivious transfer* (OT) [47], Brassard, Crépeau and Robert [24] note that it should be impossible for a receiver to gain joint information on several secrets held by a 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 breach the security of 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 frequency 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 [51] and *Kasiski method* [66]). Aware of this weakness, OT and *distributed oblivious transfer* (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. [20, 74], it is claimed that 'a party cannot learn more than a linear combination of secrets'. In this chapter, it is shown that such a claim is 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, as shown in the following basic example.

Example 3.1. *Let ω_1 and ω_2 be two secrets in the prime finite field $\mathbb{K} = GF(11)$ with the probability distributions:*

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

It is assumed that a party $\mathcal{A}$ 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, $\mathcal{A}$ just has to compare ℓ with the values 0, 3, 10 and 2 to determine the secrets ω_1 and ω_2.

Using this property, it is demonstrated that depending on the probability distributions of the secrets held by a sender, polynomial interpolation-based DOT protocols may allow a receiver to obtain all secrets. In other words, in this type of protocol, if a coalition of $t > 0$ curious servers obtains a linear combination of secrets, security condition C_3 (security

conditions and parameters are defined in Sect. 2.2.3) cannot be satisfied for $\lambda_S \geq t$ and similarly, if a curious or malicious receiver obtains a linear combination of secrets, security condition $C_{4.1}$ cannot be satisfied.

In addition, a demonstration is given showing that in Blundo, D'Arco, De Santis, and Stinson's sparse polynomial interpolation-based DOT protocol [20], 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 settings of the different DOT protocols described hereafter encompass a sender $\mathcal{S}$ who owns n secrets $\omega_1, \omega_2, \ldots, \omega_n$ ($n > 1$) selected in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime), a receiver $\mathcal{R}$ who wishes to learn a secret ω_σ, and m servers S_j ($j \in \mathcal{I}_m \subseteq \mathbb{K}^*$, $|\mathcal{I}_m| = m$, and $m > 1$). It is assumed that all operations are executed in $\mathbb{K}$ and that $p > \max(n, m)$. The minimum number of servers the receiver has to contact to obtain ω_σ is k ($1 < k \leq m$).

The organization of the chapter is as follows. In Sect. 3.2 it is shown 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. 3.3, the general form of polynomial interpolation-based DOT protocols is shortly described, as well as the technique allowing a receiver to obtain a linear combination of secrets. Section 3.4 is devoted to the analysis of some protocols [8, 19, 74, 78] in the light of the previous section. In Sect. 3.5, it is proved that even Blundo et al.'s sparse polynomial interpolation-based DOT protocol [20], designed to protect the sender's privacy against a malicious receiver in presence of honest servers, does not satisfy security condition $C_{4.1}$. A conclusion follows in Sect. 3.6.

3.2 Linear Combination of Secrets

It is not difficult to generalize the example given in the previous section. Hence, the following lemma is obtained:

Lemma 3.2. *Let $\omega_1, \omega_2, \ldots, \omega_n$ be n secrets in the finite field $\mathbb{K}$ and s be the integer such that $2^s \leq p < 2^{s+1}$. If $n \leq s$ and $\ell \in \mathcal{V} = \{0, 1, \ldots, 2^n - 1\} \subseteq \mathbb{K}$, there exist 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 $\ell = \omega_1 + \omega_2 + \cdots + \omega_n$.*

Proof.

Given an element ℓ of $\mathbb{K}$ such that $0 \leq \ell \leq 2^n - 1$ (if $a_{\mathbb{K}} \in \mathbb{K}$ and $b_{\mathbb{K}} \in \mathbb{K}$, where $\mathbb{K}$ is a prime finite field, by abuse of language we write $a_{\mathbb{K}} \leq b_{\mathbb{K}}$ if the corresponding elements of $\mathbb{N}$, $a_{\mathbb{N}}$ and $b_{\mathbb{N}}$ are such that $a_{\mathbb{N}} \leq b_{\mathbb{N}}$), we just have to exhibit n probability distributions $p_{\omega_1}, p_{\omega_2}, \ldots, p_{\omega_n}$, such that only one n-tuple $\boldsymbol{v} = [\, \omega_1, \omega_2, \ldots, \omega_n \,]$ allows the linear combination $\ell = \omega_1 + \omega_2 + \cdots + \omega_n$ to be satisfied.

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

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

If $\ell \in \mathcal{V}$, let $B^{(\ell)} = b_{n-1}^{(\ell)} b_{n-2}^{(\ell)} \ldots b_1^{(\ell)} b_0^{(\ell)}$ be the unique binary representation of ℓ. The n-tuple $\left[\, b_{n-1}^{(\ell)}, b_{n-2}^{(\ell)}, \ldots, b_1^{(\ell)}, b_0^{(\ell)} \,\right]$ is denoted by $\beta^{(\ell)}$. Then, assuming that the set of n-tuples $\{\, \beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(2^n-1)} \,\}$ is denoted $\mathcal{U}$, we define the function

$$\begin{aligned} f : \mathcal{V} &\longrightarrow \mathcal{U} \\ \ell &\longmapsto \beta^{(\ell)} . \end{aligned}$$

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 exists a unique n-tuple $\left[\, b_{n-1}^{(\ell)}, b_{n-2}^{(\ell)}, \ldots, b_1^{(\ell)}, b_0^{(\ell)} \,\right]$ where $b_i^{(\ell)} \in \{\, 0, 1 \,\}$ for $i = 0, 1, \ldots, n-1$ such that ℓ is written as a linear combination of the secrets $\omega_1, \omega_2, \ldots, \omega_n$:

$$\begin{aligned} \ell &= b_{n-1}^{(\ell)} \times 2^{n-1} + b_{n-2}^{(\ell)} \times 2^{n-2} + \cdots + b_1^{(\ell)} \times 2^1 + b_0^{(\ell)} \times 2^0 \\ &= \omega_n + \omega_{n-1} + \cdots + \omega_2 + \omega_1 \end{aligned}$$

We conclude that $\omega_i = b_{i-1}^{(\ell)} \times 2^{i-1}$ for $i = 1, 2, \ldots, n$. □

Using this basic result, in the next section some polynomial interpolation-based DOT protocols are reviewed and it it shown that their security is weaker than expected.

A more formal approach to prove that the knowledge of a linear combination of secrets may give information on the secrets themselves could be based on the theory of information (see Sect. 2.1.3). Let $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ be two random variables associated with the secrets ω_1 and ω_2 respectively and $\boldsymbol{L}$ the random variable associated with the linear combination ℓ. It is sufficient to find a joint probability distributions $p_{\Omega_1\Omega_2}$ such that for each secret, the conditional entropy of the secret given the linear combination is smaller than the entropy of the secret (i.e., $H(\boldsymbol{\Omega}_1 \mid \boldsymbol{L}) < H(\boldsymbol{\Omega}_1)$ and $H(\boldsymbol{\Omega}_2 \mid \boldsymbol{L}) < H(\boldsymbol{\Omega}_2)$). Examples of such a distribution are given in App. B, both for dependent and independent random variables.

3.3 Polynomial Interpolation-based Distributed Oblivious Transfer Protocols

Each of the existing unconditional secure polynomial interpolation-based DOT protocols, e.g. [8, 19, 20, 29, 74, 78], follows the same principle (security parameters λ_R, λ_S, and λ_C are described in Sect. 2.2.3).

- Before the protocol is executed, some details are made public: the number n of secrets, the threshold parameter k, the sender's privacy parameters λ_S and 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 $E\colon [\![1, n]\!] \longrightarrow \mathbb{K}^e$ 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 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 $\mathcal{I}_t \subseteq \mathcal{I}_m$ 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 $\boldsymbol{v}_j = \left[\, V_1(\boldsymbol{z}_j), V_2(\boldsymbol{z}_j), \ldots, V_N(\boldsymbol{z}_j) \,\right]$.

  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 $\left[\, i_j, \boldsymbol{u}_j \bullet \boldsymbol{v}_j \,\right]$ and to calculate $\omega_\sigma = R(0)$ (see Sect. 2.1.1.3).

The characteristics of the sparse polynomial interpolation-based DOT protocols analysed hereafter ([8, 19, 20, 74, 78]) are described in App. C.

Note that if the receiver $\mathcal{R}$ does not follow the protocol and, instead of $E(\sigma)$, prepares 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

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

Consequently, if $\mathcal{R}$ is able to determine an e-tuple $[\,\gamma_1, \gamma_2, \ldots, \gamma_e\,]$ such that the equality $\sum_{i=1}^{N} \alpha_i a_{i,0} = 0$ is satisfied, 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 for $j = 1, 2, \ldots, n$, $\sum_{i=1}^{N} \alpha_i a_{i,j} = 1$, then $\mathcal{R}$ is able to establish the environment of the scenario of Sect. 3.2.

3.4 Weaknesses of Some Distributed Oblivious Transfer Protocols

3.4.1 Protocols Insecure Against Curious Servers

In 2000, Naor and Pinkas [74] 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. 3.3 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 App. C.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 3.2) determine both secrets. Consequently, for any value of λ_S ($1 \leq \lambda_S \leq m$), security condition C_3 is not guaranteed.

The sparse polynomial interpolation-based unconditionally secure DOT protocol proposed by Blundo et al. [19] in 2002 is an extension of Naor and Pinkas' protocol to n secrets $\omega_1, \omega_2, \ldots, \omega_n$ where $n > 1$. 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 ($1 < i \leq n$) is $U_i(0) = \omega_i - \omega_1$ (see App. C.2.1). 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 and thus holds linear combinations of the secrets. Again, according to Lemma 3.2, each server may determine all the secrets of the sender and, for any value of λ_S ($1 \leq \lambda_S \leq m$), security condition C_3 is not satisfied.

Another polynomial interpolation-based unconditionally secure DOT protocol was introduced in 2002 by Nikov et al. [78]. 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 App. C.3). Note that if $\lambda_S = \lambda_C = 0$, which is not allowed with the security model presented in Sect. 2.2.3 since $\lambda_S \geq 1$, 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 3.2, 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 [8], 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 App. C.4). Again, with the security model presented in Sect. 2.2.3, for any value of λ_S ($1 \leq \lambda_S \leq m$), security condition C_3 is not guaranteed.

3.4.2 Protocols Insecure Against a 'Greedy' Receiver

In the sparse polynomial interpolation-based unconditionally secure DOT protocol introduced by Naor and Pinkas [74], the encoding function is $E(\sigma) = [\, 1, \delta_1^\sigma \,]$ 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 to each server S_j ($j \in \mathcal{I}_t$) a request $z_j = [\, 1, 1/2 \,]$ (because the first term is constant and public, it is not included in the request), it receives back $\sum_{i=1}^{2} U_i(j) V_i(z_j) = U_1(j) + 1/2(\omega_2 - \omega_1)$.

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 3.2) determine both secrets. Consequently, security condition $C_{4.1}$ is not guaranteed.

The insecurity is the same in Blundo et al.'s protocol [19]; the receiver sends the n-tuple request $z_j = [\,1, 1/n, 1/n, \ldots, 1/n\,]$ to each server S_j ($j \in \mathcal{I}_t$). The linear combination determined by the receiver is then $R(0) = 1/n \sum_{i=1}^{n} \omega_i$. As in Naor and Pinkas' protocol, security condition $C_{4.1}$ is not guaranteed.

In the DOT protocol introduced by Nikov et al. [78], 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 as in Blundo et al.'s protocol. Therefore, with the same n-tuple request $z_j = [\,1, 1/n, 1/n, \ldots, 1/n\,]$ 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. [8] 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.

3.5 A More Robust Protocol

Blundo et al. [20] ameliorated the protocol presented in [19] to prevent (*a*) the servers and (*b*) the receiver from learning a linear combination of secrets. It is shown below that in spite of three different improvements, the protocol is still insecure regarding a 'greedy' receiver.

3.5.1 First Improvement

To prevent servers from receiving a linear combination of secrets (see Sect. 3.4.1), 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

- 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 $\mathcal{S}$ 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 [89], of $r_2, r_3, \ldots, r_n$ (see Sect. 2.3.1).

In the transfer phase, the receiver $\mathcal{R}$ selects the index $\sigma \in [\![1, n]\!]$ of the secret she wishes to obtain as well as a set $\mathcal{I}_k \subseteq \mathcal{I}_m$ 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 $\boldsymbol{z}_j = [\, 1, Z_2(j), Z_3(j), \ldots, Z_n(j) \,]$, it calculates $\boldsymbol{v}_j = \boldsymbol{z}_j$ and returns to $\mathcal{R}$ not only $\boldsymbol{u}_j \cdot \boldsymbol{v}_j$ 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.

Note that (a) the masks r_i ($2 \leq i \leq n$) may be nil since they are selected in $\mathbb{K}$ and (b) the secret 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 as shown in the following example.

Example 3.3. *In the finite field $\mathbb{K} = GF(11)$, it is assumed 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 $r_2\omega_2 = \omega_1$ and $r_2 \neq 0$ ($r_2 \in \mathbb{K}^*$ as 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 $\mathbb{K}^*$.

It is observed that if the use of shared 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 3.2). 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 to 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 ([20], Sect. 5, p. 350):

> 'Notice that, in [74], 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 3.2, the knowledge of a linear combination of secrets and of the probability distribution of the secrets may lead to the knowledge of all secrets.

3.5.2 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 [74], Blundo et al. have modified the protocol so that it is executed twice: at first on masks randomly selected in $\mathbb{K}$ (uniform distribution) and secondly 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 App. C.2.2.

It is observed 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 App. F).

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 power such that $q \geq np$, the field GF(p) could be replaced with a field GF(q) and the pad $(i-1) \times p$ added to secret ω_i $(1 \leq i \leq n)$. It is noted 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 [20], is not satisfied) against a 'greedy' receiver. The case is illustrated with the same example (App. F), 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 $c_1 = \frac{-\alpha_2}{\alpha_1} c_2$ 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 to be 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. 3.5.1). 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, assuming that $R^{(2)}(0) = 0$, the receiver can infer $c_1 = \frac{-\alpha_2 r_2^{(2)}}{\alpha_1 r_1^{(2)}} c_2$ which, reported in the first equation gives $\alpha_2 c_2 (r_2^{(1)} \omega_2 - \frac{r_1^{(1)} r_2^{(2)}}{r_1^{(2)}} \omega_1) = R^{(1)}(0)$. 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 combinations of masked secrets (which is the result of the first improvement) and linear combinations of random masks. To conclude, only the second improvement of the protocol is necessary.

3.5.3 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 ([20], 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 $r_2^{(1)}, r_3^{(1)}, \ldots, r_n^{(1)}$.'

It is shown 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 3.4. *In the finite field $\mathbb{K} = GF(5)$, it is assumed that $n = 2$ and that the probability distributions of ω_1 and ω_2 are:*

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

Since the secrets are masked only once (see previous section), it can also be assumed 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 $\boldsymbol{u}_j^{(1)} = \left[U_1^{(1)}(j), U_2^{(1)}(j) \right]$ (for the masked secrets) as well as the pair $\boldsymbol{u}_j^{(2)} = \left[U_1^{(2)}(j), U_2^{(2)}(j) \right]$

(for the masks). The server S_j is able to calculate $d_j = U_1^{(1)}(j) - U_1^{(2)}(j) = c_1(\omega_1 - 1)$. It is assumed 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 (for any value of λ_S ($1 \leq \lambda_S \leq m$), 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 transfer phase the receiver sends the request $\boldsymbol{z}_i = [\,1, 0\,]$ to $k-1$ servers S_i ($i \in \mathcal{I}_{k-1} \subseteq \mathcal{I}_k$, $|\mathcal{I}_{k-1}| = k-1$) 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 (the polynomial $\sum_{i=1}^{k-1} a_i x^i$ is denoted by $P(x)$):

- First, as soon as $\mathcal{R}$ has received a response $\left[\, \boldsymbol{u}_j^{(1)} \bullet \boldsymbol{v}_j, \boldsymbol{u}_j^{(2)} \bullet \boldsymbol{v}_j \,\right]$, where $\boldsymbol{v}_j = \boldsymbol{z}_j$, from a server S_j ($j \in \mathcal{I}_{k-1}$), 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 $P' \in \mathbb{K}_{k-2}[X]$. With $k-1$ values ${}^1/_j P(j)$, the polynomial P' may be interpolated, which allows $\mathcal{R}$ to compute $P = xP'$.

- Finally, from the server S_ℓ, $\mathcal{R}$ obtains

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

  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.

3.6 Conclusion

It was shown that when a party is able to obtain a linear combination of secrets, the sender's privacy (security conditions C_3 for any value of λ_S ($1 \leq \lambda_S \leq m$) and $C_{4.1}$) may not be guaranteed for all distributions of secrets. As a consequence, some weaknesses have been identified in the following polynomial interpolation-based DOT protocols:

- Naor and Pinkas' sparse polynomial interpolation-based protocol [74]: without the technique described in Sect. 4, p. 214, security condition C_3 for any value of λ_S ($1 \leq \lambda_S \leq m$), and security condition $C_{4.1}$ are not guaranteed (in particular, Theorem 1 in [74] 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 [19] which extends to n secrets Naor and Pinkas' protocol.

- Nikov et al.'s protocol [78] and Beimel et al.'s protocol [8]: for some values of the protocols' parameters, security condition C_3 is not guaranteed (for any value of λ_S ($1 \leq \lambda_S \leq m$)). 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 [78] cannot guarantee security condition $C_{4.1}$, in spite of the claim of the designers.

- Blundo et al.'s protocol [20]: 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' 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.

It was also observed that the DOT protocol introduced in 2009 by Cheong, Koshiba, and Yoshiyama [29] 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' protocol.

Finally, note that in the DOT protocols introduced in the next chapters, the receiver cannot obtain a linear combination of secrets: for example, protocols in Chaps. 5, 6, and 7 use the technique proposed by Naor and Pinkas [74], and the protocol described in Chap. 8 has an inherent protection, due to the use of a broadcast channel.

In the next chapter is proposed a DOT protocol satisfying Blundo et al.'s four security conditions B_1, B_2, B_3, and B_4 [20] (see Sect. 1.3.2). One of the components of the protocol is a specific mechanism designed to prevent the receiver from obtaining a linear combination of secrets.

4

A Strongly Secure Polynomial Interpolation-based Distributed Oblivious Transfer Protocol

In an ideal *distributed oblivious transfer* where the receiver must contact k servers to learn one of the secrets held by the sender, security and privacy should be guaranteed against a coalition of less than k parties. However, Blundo, D'Arco, De Santis, and Stinson [20] have demonstrated that this level of security is impossible to achieve in one-round polynomial interpolation-based constructions.

In this chapter, it is shown that if communication is allowed amongst the servers, it is possible to design an unconditionally secure, polynomial interpolation-based distributed oblivious transfer protocol with the highest level of security. More precisely, in the resulting construction, a receiver who contacts k servers and is able to corrupt up to $k-1$ servers (not necessarily from the set of the contacted servers) cannot learn more than one secret.

4.1 Introduction

In the original sparse polynomial interpolation-based *distributed oblivious transfer* (DOT) protocol [74] introduced by Naor and Pinkas, as well as in its generalization [19, 20] presented by Blundo, D'Arco, De Santis, and Stinson, 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, while the sender cannot learn the receiver's choice.

A simplified overview of the DOT protocol presented in [20], allowing a receiver to obtain one of the n secrets of the sender, is described in Sect. 2.3.2. As pointed out by Blundo et al. [19, 20], their protocol is *private*, that is, it satisfies security conditions C_1, C_2 with $\lambda_R = k - 1$, C_3 with $\lambda_S = k - 1$ and $C_{4.1}$ (security conditions and parameters are defined in Sect. 2.2.3). However, the protocol's security is not *strong*, because it does not satisfy condition $C_{4.2}$ with $\lambda_C = k - 1$ (the receiver colluding with only one corrupt server can determine all the secrets of the sender). In fact, Blundo et al. have proved that security condition $C_{4.2}$ with $\lambda_C = k - 1$ cannot be guaranteed with a one-round DOT protocol — a round being defined as a set of consistent requests/responses exchanged between the receiver and k servers — when security condition C_2 with $\lambda_R = k - 1$ is satisfied.

A simple observation is that, in polynomial interpolation-based DOT protocols, the amount of information given to each server is too high. A fundamental requirement for achieving condition $C_{4.2}$ with $\lambda_C = k - 1$ is that not only each server, but also any coalition of up to $k - 1$ servers, should not gain a linear combination of any of two secrets (see Chap. 3). To meet this requirement, a solution consists, for the sender, in storing the secret values in the servers using a (k, m)-threshold secret sharing scheme (see Sect. 4.4.1).

In this chapter, it is shown that allowing communication amongst the servers enables the design of a one-round, polynomial interpolation-based, unconditionally and strongly secure DOT protocol (the protocol is private and satisfies condition $C_{4.2}$ with $\lambda_C = k - 1$).

Note that allowing communication amongst the servers is a silent condition in some DOT protocols [19, 20, 74]. Indeed, since given the transcript of the interaction with k servers, a coalition of the receiver and only one dishonest server is able to learn all secrets, there must be a mechanism for ensuring that the receiver does not contact more than k servers (see technique proposed by Naor and Pinkas [74]). This kind of mechanism could be implemented with communication amongst the servers.

The organization of the chapter is as follows. Definitions and notations are provided in Sect. 4.2. Section 4.3 briefly describes the model related to the protocol introduced in the chapter. The components involved in the system are presented in Sect. 4.4. Section 4.5 is devoted to the detailed description of the protocol. In Sect. 4.6, the proposed scheme is evaluated and it is demonstrated that it guarantees security conditions C_1, C_2 for $\lambda_R = k-1$, C_3 for $\lambda_S = k - 1$, and C_4 for $\lambda_C = k - 1$. A conclusion follows in Sect. 4.7.

4.2 Preliminaries

The setting of the protocol is similar to the setting of DOT protocols in [19, 20, 74], i.e., it encompasses a sender $\mathcal{S}$ who owns n secrets $\omega_1, \omega_2, \ldots, \omega_n$ $(n > 1)$ selected in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power), a receiver $\mathcal{R}$ who wishes to learn a secret ω_σ

($\sigma \in \mathcal{I}_n = [\![1, n]\!]$), and m servers S_i ($i \in \mathcal{I}_m \subseteq \mathbb{K}^*$, $|\mathcal{I}_m| = m$, $m > 1$). It is assumed that all operations are executed in $\mathbb{K}$ and that $p > \max(n-1, m)$. The receiver $\mathcal{R}$ has to contact at least k servers to obtain ω_σ ($1 < k \leq m$).

The protocol is based on polynomial interpolation. Therefore, a few definitions and theorems related to polynomials and quasi-random polynomials (see Sect. 2.1.1.2) are given.

Definition 4.1. *Let $\mathbf{\Phi} = [\, F_1, F_2, \ldots, F_n \,]$ be an n-tuple of polynomials of $\mathbb{K}_t[X]$, where $t \in \mathbb{N}$.*

The n-tuple $\mathbf{\Phi}(x) = [\, F_1(x), F_2(x), \ldots, F_n(x) \,]$ is defined as the n-tuple of corresponding polynomial functions. For any element $\alpha \in \mathbb{K}$, the n-tuple of polynomial functions $F_1(x), F_2(x), \ldots, F_n(x)$ evaluated for the argument $x = \alpha$ is denoted $\mathbf{\Phi}_\alpha = \mathbf{\Phi}(\alpha) = [\, F_1(\alpha), F_2(\alpha), \ldots, F_n(\alpha) \,]$. By homogeneity, the value of a polynomial function F applied to the argument $x = \alpha$ is denoted by F_α.

Definition 4.2. *Let $\mathbf{\Phi} = [\, F_1, F_2, \ldots, F_n \,]$ and $\mathbf{\Omega} = [\, Z_1, Z_2, \ldots, Z_n \,]$ be two n-tuples of polynomials of $\mathbb{K}[X]$. The dot product operation (denoted $\bullet$) between $\mathbf{\Phi}$ and $\mathbf{\Omega}$ is defined as*

$$\mathbf{\Phi} \bullet \mathbf{\Omega} = \sum_{i=1}^{n} F_i \times Z_i \,.$$

Lemma 4.3. *Let F be a quasi-random polynomial of $\mathbb{K}_k[X]$ ($k > 1$), with a constant coefficient $\alpha \in \mathbb{K}$, and G a (quasi-random) polynomial of $\mathbb{K}_k[X]$, with a constant coefficient $\beta \in \mathbb{K}$. Then, $H = F + G$ is a quasi-random polynomial of $\mathbb{K}_k[X]$, with a constant coefficient $\alpha + \beta$.*

Proof.

The degree of the sum of two polynomials is lower or equal to the maximum degree of the two original polynomials. Thus $\deg(H) \leq \max(\deg(F), \deg(G)) \leq k$. It follows $H \in \mathbb{K}_k[X]$.

The quasi-random polynomial F, the (quasi-random) polynomial G, and their sum, H, may be written under the form:

$$F = \sum_{i=0}^{k} f_i X^i, \quad G = \sum_{i=0}^{k} g_i X^i, \quad F + G = \sum_{i=0}^{k} (f_i + g_i) X^i, \quad \text{and} \quad H = \sum_{i=0}^{k} h_i X^i,$$

where $f_0 = \alpha$, $f_i \in_R \mathbb{K}$ ($1 \leq i \leq k$), $g_0 = \beta$, $g_i \in_R \mathbb{K}$ if G is quasi-random and $g_i \in \mathbb{K}$ otherwise ($1 \leq i \leq k$), and $h_i \in \mathbb{K}$. The $(k+1)$-tuple of polynomials $\left[\, 1, X, X^2, \ldots, X^k \,\right]$ is a basis of the vector space $\mathbb{K}_k[X]$; therefore, since $H = F + G$, we infer $h_i = f_i + g_i$ for $i = 0, 1, \ldots, k$. For $1 \leq i \leq k$, the coefficient $f_i + g_i$ is random (the sum of a random element of $\mathbb{K}$ with another element of $\mathbb{K}$ is random) and the constant term $h_0 = f_0 + g_0 = \alpha + \beta$ has a predetermined value. Thus, H is a quasi-random polynomial. □

Theorem 4.4 (Addition of shares)**.** *Let F and G be two quasi-random polynomials of $\mathbb{K}_{k-1}[X]$ $(k > 1)$ generated by (k, m)-threshold Shamir's secret sharing schemes (see Sect. 4.4.1) to share respectively the secrets α and β of $\mathbb{K}$ amongst m parties. It is also assumed that $\mathcal{I}_m \subseteq \mathbb{K}^*$ is a set of m distinct indices. For each index $i \in \mathcal{I}_m$, the shares $F(i)$ and $G(i)$ are defined. If P is the sum of F and G, then:*

(i) P is a quasi-random polynomial of $\mathbb{K}_{k-1}[X]$,

(ii) $P(0) = \alpha + \beta$ and for $i \in \mathcal{I}_m$, the share $P(i)$ is the sum of the shares $F(i)$ and $G(i)$.

In other words, P may be considered as a polynomial generated by a (k, m)-threshold Shamir's secret sharing scheme to share the secret $\alpha + \beta$.

Proof.

(i) Immediate with the application of Lemma 4.3.

(ii) From the definition 2.2 of the additive law of composition in $\mathbb{K}[X]$, it holds $P(x) = (F+G)(x) = F(x)+G(x)$, for $x \in \mathbb{K}$. Consequently, for $x = 0$, $P(0) = F(0)+G(0)$. As F and G are the sharing polynomials for the secrets α and β, we have $F(0) = \alpha$ and $G(0) = \beta$. It follows $P(0) = \alpha + \beta$. Still from the relationship $P(x) = F(x) + G(x)$, we trivially obtain for $i \in \mathcal{I}_m$, $P(i) = F(i) + G(i)$.

□

Theorem 4.5 (Product of shares)**.** *Let F and G be two quasi-random polynomials of $\mathbb{K}_{k-1}[X]$ $(k > 1)$ generated by (k, m)-threshold Shamir's schemes to share respectively the secrets α and β of $\mathbb{K}$ amongst m parties. It is also assumed that $\mathcal{I}_m \subseteq \mathbb{K}^*$ is a set of m distinct indices and that $m \geq 2k-1$. For each index $i \in \mathcal{I}_m$, the shares $F(i)$ and $G(i)$ are defined. If Q is the product of F and G, then:*

(i) $Q \in \mathbb{K}_{2k-2}[X]$,

(ii) $Q(0) = \alpha \times \beta$ and for $i \in \mathcal{I}_m$, the share $Q(i)$ is the product of the shares $F(i)$ and $G(i)$.

Proof.

(i) If $F = \sum_{i=0}^{k-1} f_i X^i$ ($f_0 = \alpha$ and $f_i \in_R \mathbb{K}$, $1 \leq i \leq k-1$) and $G = \sum_{i=0}^{k-1} g_i X^i$ ($g_0 = \beta$ and $g_i \in_R \mathbb{K}$, $1 \leq i \leq k-1$), then using the Cauchy product form of the multiplicative law of composition in $\mathbb{K}[X]$, we have

$$Q = \sum_{\ell=0}^{2k-2} \Big(\sum_{i+j=\ell} f_i g_j \Big) X^\ell$$

where $f_i = 0$ if $i \geq k$ and $g_j = 0$ if $j \geq k$. The degree of Q is at most $2k-2$, so $Q \in \mathbb{K}_{2k-2}[X]$.

(ii) Using the definition 2.2 of the multiplicative law of composition in $\mathbb{K}[X]$, we can write $Q(x) = (F \times G)(x) = F(x) \times G(x)$ for $x \in \mathbb{K}$. In particular, $Q(0) = F(0) \times G(0) = \alpha \times \beta$. Of course, the relationship is also true for $i \in \mathcal{I}_{\mathrm{m}}$.

□

The polynomial Q cannot be considered as quasi-random because, by construction, it cannot be irreducible. However, if Q is added to a quasi-random polynomial of $\mathbb{K}_{2k-2}[X]$ whose constant term is zero, the resulting polynomial is quasi-random (see Lemma 4.3) and may be considered as a polynomial generated by a $(2k-1, m)$-threshold Shamir's scheme to share the secret $\alpha \times \beta$.

Corollary 4.6. *Let $\mathbf{\Phi} = [\, F_1, F_2, \ldots, F_n \,]$ and $\mathbf{\Gamma} = [\, G_1, G_2, \ldots, G_n \,]$ be two n-tuples of quasi-random polynomials of $\mathbb{K}_{k-1}[X]$. For $i = 1, 2, \ldots, n$, the polynomial F_i (resp. G_i) is generated by a (k, m)-threshold Shamir's scheme to share the secret φ_i (resp. the secret γ_i). It is assumed that $m \geq 2k-1$. For each index $i \in \mathcal{I}_m$, the shares $[\, F_1(i), F_2(i), \ldots, F_n(i) \,]$ and $[\, G_1(i), G_2(i), \ldots, G_n(i) \,]$ are defined. If the polynomial Q is the dot product of $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ (i.e., $Q = \mathbf{\Phi} \cdot \mathbf{\Gamma}$), then:*

(i) $Q \in \mathbb{K}_{2k-2}[X]$,

(ii) $Q(0) = \sum_{i=1}^{n} \varphi_i \times \gamma_i$ *and for* $\ell \in \mathcal{I}_m$, $Q(\ell) = \sum_{i=1}^{n} F_i(\ell) \times G_i(\ell)$.

By simplification, it is considered that the polynomial Q is quasi-random. However, to obtain a quasi-random polynomial, it would be necessary to add Q to a quasi-random polynomial of $\mathbb{K}_{2k-2}[X]$ whose constant term is zero. The resulting polynomial would be quasi-random (see Lemma 4.3) and could be considered as a polynomial generated by a $(2k-1, m)$-threshold Shamir's scheme to share the secret $\sum_{i=1}^{n} \varphi_i \times \gamma_i$.

Proof.

(i) From Theorem 4.5(i), $F_i \times G_i \in \mathbb{K}_{2k-2}[X]$ $(1 \leq i \leq n)$. Then, by generalization of Theorem 4.4(i), $\sum_{i=1}^{n} F_i \times G_i \in \mathbb{K}_{2k-2}[X]$. Consequently, $Q \in \mathbb{K}_{2k-2}[X]$.

(ii) From Theorems 4.5(ii) and 4.4(ii), it holds $Q(x) = \sum_{i=1}^{n} F_i(x) \times G_i(x)$ for $x \in \mathbb{K}$. In particular, $Q(0) = \sum_{i=1}^{n} F_i(0) \times G_i(0) = \sum_{i=1}^{n} \varphi_i \times \gamma_i$. Of course, the relationship is also true for $\ell \in \mathcal{I}_{\mathrm{m}}$, i.e., $Q(\ell) = \sum_{i=1}^{n} F_i(\ell) \times G_i(\ell)$.

□

Notations:
Addition and multiplication of n-tuples (of the same variety) are defined naturally. For example, if $\boldsymbol{U} = [\,u_1, u_2, \ldots, u_n\,]$ and $\boldsymbol{V} = [\,v_1, v_2, \ldots, v_n\,]$, then the sum of $\boldsymbol{U}$ and $\boldsymbol{V}$ is $\boldsymbol{U} + \boldsymbol{V} = [\,u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n\,]$ and the product of $\boldsymbol{U}$ and $\boldsymbol{V}$ is $\boldsymbol{U} \times \boldsymbol{V} = [\,u_1 \times v_1, u_2 \times v_2, \ldots, u_n \times v_n\,]$. Similarly, the product $\alpha \times \boldsymbol{U}$ between an element α of $\mathbb{K}$ and an n-tuple $\boldsymbol{U} = [\,u_1, u_2, \ldots, u_n\,]$ of n elements of $\mathbb{K}$ is defined as the n-tuple $[\,\alpha \times u_1, \alpha \times u_2, \ldots, \alpha \times u_n\,]$.
Each server participating in the protocol is identified by an index selected in $\mathcal{I}_{\mathrm{m}} \subseteq \mathbb{K}^*$, $|\mathcal{I}_{\mathrm{m}}| = m$. Thus, the server associated with the index $\ell \in \mathcal{I}_{\mathrm{m}}$ is identified as S_ℓ.

4.3 Model

In addition to the three categories of parties — the sender $\mathcal{S}$, the receiver $\mathcal{R}$ and the servers S_i ($i \in \mathcal{I}_{\mathrm{m}}$) — involved in the protocol presented in this chapter, an adversary whose characteristics are defined below needs to be taken into account.

Like other DOT protocols, the protocol presented in this chapter is composed of two phases: a *set-up phase* and a *transfer phase*. During the set-up step, the sender generates shares of the n secrets he holds and distributes them to the m servers. The sender does not intervene in the rest of the protocol. During the transfer phase, the receiver has to contact k servers ($1 < k \leq m$) to collect enough shares to construct ω_σ.

In the scheme, however, the inequality $m \geq 2k - 1$ must be satisfied. This is because in some steps of the protocol, $2k - 1$ distinct servers need to exchange information amongst themselves. Note that since the inequality is satisfied, security condition C_4 allows the receiver to corrupt up to $\lambda_{\mathrm{C}} = k - 1$ servers different from the k servers chosen by the receiver for gaining shares of the chosen secret.

4.3.1 The Adversary

During the execution of the DOT protocol, the receiver contacts k servers. In addition, she is allowed to corrupt up to $k - 1$ servers, possibly amongst the contacted servers. That is, the receiver plays the role of an active adversary — although the corrupted servers have a passive behaviour, i.e., they do not deviate from the protocol but share all their data with the receiver — who wishes to breach the sender's security by learning more than one secret from the protocol. As in other DOT protocols (for example, see [74]), it is assumed that a mechanism prevents the receiver from contacting more than k servers in one round.

On the other hand, up to $k - 1$ servers may collaborate to learn the choice σ of the receiver. In this scenario, the coalition of servers may be considered as an adversary who wishes to breach the privacy of the receiver.

4.3.2 Overview of the Protocol

The key idea underlying the design of the protocol is that if a sender holds an n-tuple $\mathbf{\Omega} = [\,\omega_1, \omega_2, \ldots, \omega_n\,]$ of n secrets of $\mathbb{K}$ ($n > 1$) and if a receiver wishes to learn the secret ω_σ ($\sigma \in \mathcal{I}_\mathrm{n}$), then the receiver contributes with an n-tuple $\mathbf{\Delta} = [\,\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma\,]$, and the servers respond with the dot product of these two n-tuples, i.e., $\mathbf{\Omega} \cdot \mathbf{\Delta} = \sum_{i=1}^{n} \omega_i \times \delta_i^\sigma$, which is the requested secret.

To guarantee the security of the sender and the privacy of the receiver, the n-tuples involved in the dot product are shared thanks to Shamir's secret sharing schemes [89]. That is, the sender transmits to each server S_ℓ ($\ell \in \mathcal{I}_\mathrm{m}$) an n-tuple $\boldsymbol{u}_\ell = [\,F_1(\ell), F_2(\ell), \ldots, F_n(\ell)\,]$ of n shares, where F_i ($1 \leq i \leq n$) is the sharing polynomial related to ω_i. In the same way, to obtain a secret ω_σ, the receiver selects a subset $\mathcal{I}_\mathrm{k} \subseteq \mathcal{I}_\mathrm{m}$ of k distinct indices, sends to each server S_j ($j \in \mathcal{I}_\mathrm{k}$) an n-tuple $\mathbf{\Phi}_j = [\,Z_1(j), Z_2(j), \ldots, Z_n(j)\,]$ of n shares (Z_i is the sharing polynomial related to δ_i^σ) and receives k shares of the chosen secret ω_σ. The shares are associated with a polynomial R of $\mathbb{K}_{k-1}[X]$ and so, by interpolation, the receiver is able to determine R and to calculate the chosen secret $\omega_\sigma = R(0)$ (see Sect. 2.1.1.3).

4.4 Components of the System

The protocol is mainly based on three components.

The first one is a secret sharing scheme, allowing on one hand the sender to generate and distribute shares of the secrets he holds, and on the other hand the receiver to generate and distribute shares of the identifier of the secret she wishes to learn.

The second one is a mechanism which enables a set of parties to redistribute a secret to another set of parties. This component requires the availability of private unicast communication channels between any two parties involved in the protocol. It is assumed 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 (see Sect. 2.2.1).

The third component is a scheme which allows a set of parties to verify that the shares — generated by a secret sharing scheme — that they have received correspond to one of the values in a predefined set of values. This component also requires private communication channels between any two parties.

4.4.1 Secret Sharing Scheme

The secret sharing scheme underlying the DOT protocol presented in this chapter is a (k, m)-threshold Shamir's secret sharing scheme [89]. The scheme allows a dealer to distribute shares of a secret ω to m parties so that ω can be reconstructed by any set of k

($1 < k \leq m$) or more of these parties. The secret as well as the shares are elements of a finite field $\mathbb{K} = \mathrm{GF}(p)$. The detailed description of the scheme is given in Sect. 2.3.1.

In a semi-honest security model, Shamir's scheme is unconditionally secure; the secret ω is always reconstructed from k or more shares and the knowledge of $k-1$ or fewer shares leaves ω completely undetermined (see information-theoretic demonstration in App. A).

4.4.2 Transformation from One to Another Threshold Scheme

There are many applications that require redistribution of a secret from one set of parties to another set of parties, with possibly a different threshold. Desmedt and Jajodia [42] have considered this problem and proposed some protocols. The specific case of the reduction of the threshold is commonly discussed in multi-party computation studies. Indeed, if two secrets are shared (Shamir's scheme) thanks to two polynomials P_1 and P_2 of $\mathbb{K}_{k-1}[X]$, then from the polynomial $Q = P_1 \times P_2$ can be generated shares of the product of the two secrets (see Theorem 4.5). But the degree of Q (up to $2k-2$) corresponds to a threshold $2k-1$, and so a degree reduction is necessary to obtain a sharing polynomial corresponding to a threshold k. Such a reduction was introduced, for example, by Ben-Or, Goldwasser, and Wigderson [14].

Below is presented a combined method, allowing a set of k_1 or more parties holding shares generated by a (k_1, m_1)-threshold Shamir's scheme to distribute them under the form of shares generated by a (k_2, m_2)-threshold Shamir's scheme, where $k_1 \geq k_2$, to a set of m_2 parties.

Let ω be a secret in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power), $\mathcal{U}$ (resp. $\mathcal{V}$) be a set of $m_1 > 1$ parties U_i (resp. $m_2 > 1$ parties V_j) where $i \in \mathcal{I}_{m_1} \subseteq \mathbb{K}^*$, $\left|\mathcal{I}_{m_1}\right| = m_1$ (resp. $j \in \mathcal{I}_{m_2} \subseteq \mathbb{K}^*$, $|\mathcal{I}_{m_2}| = m_2$). It is assumed that ω is shared amongst the parties of the group $\mathcal{U}$ thanks to a polynomial F generated in a (k_1, m_1)-threshold scheme ($1 < k_1 \leq m_1$). Each party U_i holds the share $F(i)$. Let $\mathcal{I}_{k_1} \subseteq \mathcal{I}_{m_1}$ be a set of k_1 indices; it is assumed that the parties U_ℓ ($\ell \in \mathcal{I}_{k_1}$) wish to redistribute the secret ω over the set of parties $\mathcal{V}$ using a (k_2, m_2)-threshold scheme ($1 < k_2 \leq m_2$). Note that $\mathcal{U}$ and $\mathcal{V}$ are two arbitrary sets of parties. Each party U_ℓ ($\ell \in \mathcal{I}_{k_1}$) generates a polynomial G_ℓ such that

$$G_\ell = \sum_{r=1}^{k_2-1} g_{\ell,r} X^r + F(\ell) \times L_\ell(0),$$

where $g_{\ell,r} \in_R \mathbb{K}$ ($1 \leq r \leq k_2 - 1$) and L_ℓ is the Lagrange basis polynomial

$$L_\ell = \prod_{\substack{j \in \mathcal{I}_{k_1} \\ j \neq \ell}} \frac{X - j}{\ell - j}.$$

A variant of the technique described by Beaver [6] is actually applied. With the original technique, to generate a polynomial G_ℓ of $\mathbb{K}_{k_2-1}[X]$ and not $\mathbb{K}_{k_1-1}[X]$ (when $k_2 < k_1$), each party U_ℓ ($\ell \in \mathcal{I}_{k_1}$) truncates the Lagrange basis polynomial $L_\ell \in \mathbb{K}_{k_1}[X]$ thanks to a modulo X^{k_2} operation. The variant presented here is more efficient since only the free coefficient $L_\ell(0)$ needs to be calculated.

Then, for all $j \in \mathcal{I}_{m_2}$, the party U_ℓ ($\ell \in \mathcal{I}_{k_1}$) privately sends the value $G_\ell(j)$ to the party V_j. Once a party $V_j \in \mathcal{V}$ obtains k_1 partial shares, sent by the parties U_ℓ, V_j computes a new share $\rho_j = \sum_{\ell \in \mathcal{I}_{k_1}} G_\ell(j)$ as his or her share of the secret ω.

With this method, the original sharing polynomial F is replaced with a sharing polynomial F' such that

$$\begin{aligned} F' &= \sum_{\ell \in \mathcal{I}_{k_1}} G_\ell \\ &= \sum_{\ell \in \mathcal{I}_{k_1}} \left(\sum_{r=1}^{k_2-1} g_{\ell,r} X^r + F(\ell) \times L_\ell(0) \right) \\ &= \sum_{\ell \in \mathcal{I}_{k_1}} \sum_{r=1}^{k_2-1} g_{\ell,r} X^r + \sum_{\ell \in \mathcal{I}_{k_1}} F(\ell) \times L_\ell(0) \\ &= \sum_{\ell \in \mathcal{I}_{k_1}} \sum_{r=1}^{k_2-1} g_{\ell,r} X^r + F(0)\,. \end{aligned}$$

The degree of the polynomial F' is at most $k_2 - 1$ and $F'(0) = F(0) = \omega$. Therefore, k_2 parties V_j, where $j \in \mathcal{I}_{k_2} \subseteq \mathcal{V}$, $|\mathcal{I}_{k_2}| = k_2$ are able from the shares $\rho_j = F'(j)$ to interpolate F' and to determine the secret ω.

Theorem 4.7. *If $k_2 \leq k_1$, the proposed protocol for transforming a (k_1, m_1)-threshold Shamir's scheme into a (k_2, m_2)-threshold Shamir's scheme is secure.*

Proof.

To demonstrate the security of ω, we show that after the new shares have been obtained, no coalition of $k_2 - 1$ parties or fewer can infer any information about ω.

After the redistribution of shares, a coalition of $k_2 - 1$ parties holds at most $k_2 - 1$ shares $F(j)$ and $k_2 - 1$ shares $F'(j)$. We analyze the worst case, i.e., when all the members of the coalition belong to $\mathcal{U}$. First, the coalition cannot interpolate F and calculate $F(0)$ since k_1 shares $F(j)$ would be necessary, and the coalition only holds $k_2 - 1 \leq k_1 - 1 < k_1$ shares. Second, with only $k_2 - 1$ values $F'(j)$, and F' being a polynomial of $\mathbb{K}_{k_2-1}[X]$, the coalition can neither interpolate F' nor calculate $F'(0)$. Consequently, a coalition of $k_2 - 1$ parties $U_i \in \mathcal{V}$ is unable to gain any information on the secret ω. □

4.4.3 Multi-party Computation Membership of a Public Set

Suppose that in a polynomial interpolation-based DOT protocol, a malicious receiver prepares a request $\boldsymbol{q} = [\,\gamma_1, \gamma_2, \ldots, \gamma_n\,]$ of n elements of $\mathbb{K}$ while the protocol requires that she prepares an n-tuple $[\,\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma\,]$ where σ is the index of one of the secrets $\omega_1, \omega_2, \ldots, \omega_n$. In Chap. 3, it was shown that in this scenario, the receiver may learn a linear combination of the secrets, and possibly the secrets themselves. To detect a cheating receiver, a method is proposed here. This method allows the contacted servers to control that the n-tuples they have received correspond to a legitimate request, without revealing the receiver's choice, of course.

Similar private set operation protocols like *private equality test*, *private set intersection* or *private set membership* have been proposed in distributed environments [50, 70, 79]. With private equality test, a party $\mathcal{A}$ sends shares, generated by Shamir's secret sharing scheme, of a secret value a to m servers. Symmetrically, a party $\mathcal{B}$ sends shares of a secret value b to the same servers. The servers calculate if $a = b$ and return shares of the result to $\mathcal{A}$ and $\mathcal{B}$. If k ($1 < k \leq m$) is the security parameter of the protocol, no coalition of $k-1$ honest but curious servers has any information on a or b. Furthermore, if $a \neq b$, $\mathcal{A}$ does not obtain any information on b and $\mathcal{B}$ does not get any information on a. In a private set intersection protocol, a party $\mathcal{A}$ holds a set $A = \{\,a_1, a_2, \ldots, a_s\,\}$ of elements of $\mathbb{K}$ and a party $\mathcal{B}$ holds a set $B = \{\,b_1, b_2, \ldots, b_t\,\}$ of elements of $\mathbb{K}$ ($s \in \mathbb{N}, t \in \mathbb{N}$). $\mathcal{A}$ sends shares of each element of A to m servers and similarly, $\mathcal{B}$ sends shares of elements of B to the same servers. Then, the servers calculate $A \cap B$ and return shares of the result to $\mathcal{A}$ and $\mathcal{B}$. Again, if k is the security parameter of the protocol, no coalition of $k-1$ servers learns any of the elements of $A \cup B$, and $\mathcal{A}$ and $\mathcal{B}$ cannot obtain any information on the secret values a_i and b_j not in $A \cap B$ ($i \in [\![\,1, s\,]\!]$, $j \in [\![\,1, t\,]\!]$). Also, coalitions of $k-1$ servers do not obtain information on $A \cap B$. A variant of this protocol allows the parties $\mathcal{A}$ and $\mathcal{B}$ to obtain the size $|A \cap B|$ of the intersection of A and B. With the set membership functionality, a party $\mathcal{A}$ holds a secret value a while a party $\mathcal{B}$ holds a set $B = \{\,b_1, b_2, \ldots, b_t\,\}$ of secrets values. Shares of $a, b_1, b_2, \ldots, b_t$ are sent to m servers which calculate if $a \in B$; the shares of the result are returned to $\mathcal{A}$ and $\mathcal{B}$. At the end of the protocol, a coalition of $k-1$ servers has no information on $B \cup \{\,a\,\}$, $\mathcal{A}$ has no information on $B \setminus \{\,a\,\}$ and if $a \notin B$, $\mathcal{B}$ has no information on a.

Actually, all these schemes are more complex than necessary. Indeed, in these protocols, the value to control is shared amongst the servers, but the potential values for the comparisons are private too. Moreover, most of the schemes are computationally secure. This is why a simple, unconditionally secure, special-purpose solution based on Shamir's scheme, in a semi-honest model, was designed. The objective is that the servers contacted

by the receiver collectively check that the n-tuples they received contain the shares of a request $\boldsymbol{q} = [\,\gamma_1, \gamma_2, \ldots, \gamma_n\,]$ where all elements $\gamma_i \in \mathbb{K}$ are nil except one.

The main idea of the protocol is that the polynomial $P = X(X-1)$ can be used to check that a value a is 0 or 1 (it is the case if $P(a) = 0$). To verify that given n values of $\{\,0, 1\,\}$, only one of those values is 1, it is sufficient to calculate their sum s. Indeed, $0 \leq s \leq n < p$ which means that if all values are nil except one, then $s = 1$.

Let $\mathcal{A}$ be a party who holds an n-tuple $\boldsymbol{q} = [\,\gamma_1, \gamma_2, \ldots, \gamma_n\,]$ of elements of a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power) and let S_i be a server such that $i \in \mathcal{I}_{\mathrm{m}} \subseteq \mathbb{K}^*$ ($|\mathcal{I}_{\mathrm{m}}| = m$, $1 < m < p$). Each element γ_i of $\boldsymbol{q}$ is shared thanks to a (k, m)-threshold Shamir's scheme. More precisely, for each element γ_i, $\mathcal{A}$ generates a quasi-random polynomial $F_i \in \mathbb{K}_{k-1}[X]$ such that $F_i(0) = \gamma_i$. It is assumed that $m \geq 2k-1$. If $\mathcal{I}_{2\mathrm{k}\,-\,1} \in \mathcal{I}_{\mathrm{m}}$ is a set of $2k-1$ servers' indices, $\mathcal{A}$ transmits to each server S_j ($j \in \mathcal{I}_{2\mathrm{k}\,-\,1}$) the n-tuple $\boldsymbol{q}_j = [\,F_1(j), F_2(j), \ldots, F_n(j)\,]$. The polynomial $X(X-1)$ is denoted by P.

To check that there exists $\sigma \in [\![\,1, n\,]\!]$ such that $\boldsymbol{q} = [\,\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma\,]$, each server S_j ($j \in \mathcal{I}_{2\mathrm{k}\,-\,1}$) applies the following procedure:

1. S_j calculates $u_{j,i} = P(F_i(j))$ for $i = 1, 2, \ldots, n$ as well as $v_j = \sum_{i=1}^{n} F_i(j)$.
2. S_j redistributes $u_{j,1}, u_{j,2}, \ldots, u_{j,n}$ and v_j to the servers S_ℓ ($\ell \in \mathcal{I}_{2\mathrm{k}\,-\,1}$) thanks to the redistribution protocol described in Sect. 4.4.2. As a result, each server S_j holds new shares $u'_{j,1}, u'_{j,2}, \ldots, u'_{j,n}$ and v'_j of respectively $\gamma_1^2 - \gamma_1, \gamma_2^2 - \gamma_2, \ldots, \gamma_n^2 - \gamma_n$, and $\sum_{i=1}^{n} \gamma_i$.
3. S_j sends its $n+1$ shares to S_ℓ ($\ell \in \mathcal{I}_{2\mathrm{k}\,-\,1}$)
4. S_j interpolates from the received values the n values $P(\gamma_i)$ ($i \in [\![\,1, n\,]\!]$) and $\sum_{i=1}^{n} \gamma_i$. Then, S_j checks that for $i = 1, 2, \ldots, n$, $P(\gamma_i) = 0$ and that $\sum_{i=1}^{n} \gamma_i = 1$, otherwise, S_i aborts the protocol because the n-tuples sent to the servers do not correspond to a valid request.

The correctness of the protocol is guaranteed because $P(F_i) = F_i(F_i - 1)$ is a polynomial of $\mathbb{K}_{2k-2}[X]$ (so $2k-1$ shares are sufficient to interpolate $P(F_i)$) and the privacy is guaranteed by the use of the redistribution protocol (see Sect. 4.4.2).

4.5 A Strongly Secure Distributed Oblivious Transfer Protocol

This section presents a new polynomial interpolation-based, one-round DOT protocol, which satisfies all security conditions, i.e., C_1, C_2 with $\lambda_{\mathrm{R}} = k-1$, C_3 with $\lambda_{\mathrm{S}} = k-1$,

Let $\mathcal{I}_{\mathrm{m}}$ ($m > 1$) be a set of m distinct elements of $\mathbb{K}^*$ and S_j a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Input The sender $\mathcal{S}$, contributes with n secrets $\omega_1, \omega_2, \ldots, \omega_n \in \mathbb{K}$.
The receiver $\mathcal{R}$ chooses an index $\sigma \in \mathcal{I}_{\mathrm{n}} = [\![1, n]\!]$, and contributes with n private values $\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma \in [\![0, 1]\!]$.

Output $\mathcal{R}$ receives ω_σ, while $\mathcal{S}$ receives nothing.

Set-up Phase – Generation of the Servers' Shares

1. $\mathcal{S}$ applies a (k, m)-threshold Shamir's scheme to each of the elements of $\mathbf{\Omega} = [\, \omega_1, \omega_2, \ldots, \omega_n \,]$. If $F_i \in \mathbb{K}_{k-1}[X]$ is the quasi-random polynomial such that $F_i(0) = \omega_i$ $(1 \leq i \leq n)$, then $\mathcal{S}$ obtains an n-tuple of polynomials $\mathbf{\Theta} = [\, F_1, F_2, \ldots, F_n \,]$.
2. $\mathcal{S}$ transmits the n-tuple of shares $\mathbf{\Theta}_\ell = [\, F_1(\ell), F_2(\ell), \ldots, F_n(\ell) \,]$ to the server S_ℓ $(\ell \in \mathcal{I}_{\mathrm{m}})$.

Figure 4.1: A Strongly Secure Distributed Oblivious Transfer Protocol (Set-up Phase)

$C_{4.1}$, and $C_{4.2}$ with $\lambda_{\mathrm{C}} = k - 1$. The protocol, depicted in Figs. 4.1 and 4.2, is composed of a set-up phase (one step) and of a transfer phase (five steps), described in the following sections.

4.5.1 Set-up Phase – Generation of the Servers' Shares

This step is straightforward. The sender, $\mathcal{S}$, distributes the shares of the secrets of the n-tuple $\mathbf{\Omega} = [\, \omega_1, \omega_2, \ldots, \omega_n \,]$ to each server S_ℓ $(\ell \in \mathcal{I}_{\mathrm{m}})$, using (k, m)-threshold Shamir's schemes. That is, for each secret ω_i $(1 \leq i \leq n)$, $\mathcal{S}$ generates a quasi-random polynomial $F_i \in \mathbb{K}_{k-1}[X]$ such that

$$F_i = \sum_{j=1}^{k-1} f_{i,j} X^j + \omega_i,$$

where $f_{i,j} \in_R \mathbb{K}$, and computes the shares of ω_i for all servers S_ℓ $(\ell \in \mathcal{I}_{\mathrm{m}})$. Then, $\mathcal{S}$ transmits to each server S_ℓ the n-tuple of shares $\mathbf{\Theta}_\ell = [\, F_1(\ell), F_2(\ell), \ldots, F_n(\ell) \,]$. The sender does not intervene in the rest of the protocol.

4.5.2 Transfer Phase Step 1 – Generation of the Receiver's Request

The receiver, $\mathcal{R}$, chooses the index σ of the secret she wishes to obtain, as well as a subset $\mathcal{I}_{\mathrm{k}} \subseteq \mathcal{I}_{\mathrm{m}}$ of k indices of the servers she intends to contact. Then, $\mathcal{R}$ applies a (k, m)-threshold Shamir's scheme to each element of the n-tuple $\mathbf{\Delta} = [\, \delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma \,]$.

Transfer Phase Step 1 – Generation of the Receiver's Request

1. $\mathcal{R}$ chooses $\sigma \in \mathcal{I}_{\mathrm{n}}$ and selects $\mathcal{I}_{\mathrm{k}} \subseteq \mathcal{I}_{\mathrm{m}}$ such that $|\mathcal{I}_{\mathrm{k}}| = k$. Then, $\mathcal{R}$ applies a (k, m)-threshold Shamir's scheme to each element of the n-tuple $\mathbf{\Delta} = [\, \delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma \,]$. If $Z_i \in \mathbb{K}_{k-1}[X]$ is the quasi-random polynomial such that $Z_i(0) = \delta_i^\sigma$ $(1 \leq i \leq n)$, then $\mathcal{R}$ obtains an n-tuple of polynomials $\mathbf{\Phi} = [\, Z_1, Z_2, \ldots, Z_n \,]$.

2. $\mathcal{R}$ transmits to the server S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ the n-tuple $\mathbf{\Phi}_\ell = [\, Z_1(\ell), Z_2(\ell), \ldots, Z_n(\ell) \,]$.

Transfer Phase Step 2 – Redistribution of the Receiver's Input

1. The contacted servers S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ select a subset $\mathcal{I}_{2\mathrm{k}\,-\,1} \subseteq \mathcal{I}_{\mathrm{m}}$ such that $|\mathcal{I}_{2\mathrm{k}\,-\,1}| = 2k - 1$ and $\mathcal{I}_{\mathrm{k}} \subseteq \mathcal{I}_{2\mathrm{k}\,-\,1}$.

2. The servers S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ redistribute the n-tuples $\mathbf{\Phi}_\ell$ to the servers S_j $(j \in \mathcal{I}_{2\mathrm{k}\,-\,1})$ thanks to the scheme described in Sect. 4.4.2. The threshold k is preserved. As a result, each server S_j holds an n-tuple $\mathbf{\Phi}'_j = \left[\, Z'_1(j), Z'_2(j), \ldots, Z'_n(j) \,\right]$ where $Z'_i \in \mathbb{K}_{k-1}[X]$ is a quasi-random polynomial such that $Z'_i(0) = Z_i(0)$ $(1 \leq i \leq n)$.

Transfer Phase Step 3 – Verification of the Receiver's Input

Each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}\,-\,1})$ applies the verification technique described in Sect. 4.4.3 to verify if the shares received in the previous step correspond to a legitimate value of $\mathbf{\Delta}$, i.e., if the shares were generated for an n-tuple $\mathbf{\Delta} = [\, \delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma \,]$, where $\sigma \in \mathcal{I}_{\mathrm{n}}$. If this is not the case (for example, because the receiver has cheated), S_ℓ aborts the protocol.

Transfer Phase Step 4 – Computation of the Requested Secret

1. Each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}\,-\,1})$ calculates the dot product $\lambda_\ell = \mathbf{\Theta}_\ell \cdot \mathbf{\Phi}'_\ell$, which is its share of the requested secret, $\mathbf{\Theta}_0 \cdot \mathbf{\Phi}_0$, related to a quasi-random polynomial Q of $\mathbb{K}_{2k-2}[X]$. Then, each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}\,-\,1})$ redistributes its share λ_ℓ to the servers S_j $(j \in \mathcal{I}_{\mathrm{k}})$ thanks to a (k, k)-threshold scheme, as described in Sect. 4.4.2.

2. Once the redistribution has been executed, each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}\,-\,1})$ holds a value $\lambda'_\ell = R(\ell)$, where R is a quasi-polynomial of $\mathbb{K}_{k-1}[X]$ such that $R(0) = Q(0)$.

Transfer Phase Step 5 – Transfer of the Requested Secret

1. S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ sends λ'_ℓ to $\mathcal{R}$.

2. From the k responses, $\mathcal{R}$ interpolates the polynomial R of $\mathbb{K}_{k-1}[X]$ and calculates $\omega_\sigma = R(0)$ (see Sect. 2.1.1.3).

Figure 4.2: A Strongly Secure Distributed Oblivious Transfer Protocol (Transfer Phase)

That is, $\mathcal{R}$ generates n quasi-random polynomials $Z_i \in \mathbb{K}_{k-1}[X]$ $(1 \leq i \leq n)$ such that

$$Z_i = \sum_{j=1}^{k-1} z_{i,j} X^j + \delta_i^\sigma ,$$

where $z_{i,j} \in_R \mathbb{K}$, and transmits to each server S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ the n-tuple of shares $\mathbf{\Phi}_\ell = [\, Z_1(\ell), Z_2(\ell), \ldots, Z_n(\ell) \,]$.

4.5.3 Transfer Phase Step 2 – Redistribution of the Receiver's Input

In this step the k contacted servers which have received, from $\mathcal{R}$, shares generated by (k, k)-threshold Shamir's schemes, redistribute them as shares considered as generated by $(k, 2k-1)$-threshold Shamir's schemes. To perform this redistribution, the servers S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ first select a set $\mathcal{I}_{2\mathrm{k}-1}$ of $2k-1$ distinct indices of $\mathbb{K}^*$ such that $\mathcal{I}_{\mathrm{k}} \subseteq \mathcal{I}_{2\mathrm{k}-1} \subseteq \mathcal{I}_{\mathrm{m}}$. Then for every secret value δ_i^σ shared thanks to a polynomial Z_i $(1 \leq i \leq n)$, they execute the redistribution protocol described in Sect. 4.4.2, and re-share that secret value from the k servers S_ℓ $(\ell \in \mathcal{I}_{\mathrm{k}})$ to the $2k-1$ servers S_j $(j \in \mathcal{I}_{2\mathrm{k}-1})$, keeping the threshold k unchanged. At the end of the distribution, the shares associated with a secret value δ_i^σ $(1 \leq i \leq n)$ are considered as generated from a new polynomial of $\mathbb{K}_{k-1}[X]$ denoted by Z'_i (also, the n-tuple of quasi-random polynomials related to the n-tuple $\mathbf{\Delta}$ of private values is denoted by $\mathbf{\Phi}' = \left[\, Z'_1, Z'_2, \ldots, Z'_n \,\right]$). Therefore, each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}-1})$ obtains an n-tuple $\mathbf{\Phi}'_\ell = \left[\, Z'_1(\ell), Z'_2(\ell), \ldots, Z'_n(\ell) \,\right]$ of shares of the elements of $\mathbf{\Delta}$.

4.5.4 Transfer Phase Step 3 – Verification of the Receiver's Input

To prevent the receiver from learning a linear combination of secrets, the contacted servers verify that the shares they have received in the previous step correspond to a legitimate request. To this end, they apply the technique described in Sect. 4.4.3. More precisely,

1. Each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}-1})$ calculates $\mathbf{\Gamma}_\ell = \mathbf{\Phi}'_\ell \times \mathbf{\Phi}'_\ell - \mathbf{\Phi}'_\ell$ as well as $\gamma_\ell = \mathbf{\Phi}'_\ell \cdot \mathbf{1}_n$. For $i = 1, 2, \ldots, n$, let G_i denote the product of the quasi-random polynomials Z'_i and $Z'_i - 1$ of $\mathbb{K}_{k-1}[X]$. Then, the n-tuple $\mathbf{\Gamma} = \mathbf{\Phi}' \times \mathbf{\Phi}' - \mathbf{\Phi}' = [\, G_1, G_2, \ldots, G_n \,]$ is composed of polynomials of $\mathbb{K}_{2k-1}[X]$ (see Theorem 4.5). If H is the polynomial such that $H = \sum_{i \in \mathcal{I}_{\mathrm{n}}} Z'_i$, then H is a quasi-random polynomial of $\mathbb{K}_{k-1}[X]$ and for $\ell \in \mathcal{I}_{2\mathrm{k}-1}$, $\gamma_\ell = H(\ell)$ (see Theorem 4.4).

2. Then, the servers S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}-1})$ redistribute the n-tuples of shares $\mathbf{\Gamma}_\ell$ to themselves thanks to the scheme described in Sect. 4.4.2, decreasing the threshold $2k-1$ to k. The share γ_ℓ is also redistributed by the servers to themselves, but the threshold k is kept unchanged. As a result, each server S_ℓ $(\ell \in \mathcal{I}_{2\mathrm{k}-1})$ holds an n-tuple $\mathbf{\Gamma}'_\ell =$

$\left[G'_1(\ell), G'_2(\ell), \ldots, G'_n(\ell) \right]$ where $G'_i \in \mathbb{K}_{k-1}[X]$ is a quasi-random polynomial, such that $G'_i(0) = G_i(0)$ $(1 \leq i \leq n)$, as well as a value $\gamma'_\ell = H'(\ell)$, where H' is a quasi-random polynomial of $\mathbb{K}_{k-1}[X]$ too, such that $H'(0) = H(0)$.

3. Each server S_ℓ $(\ell \in \mathcal{I}_{2k-1})$ transmits $\mathbf{\Gamma}'_\ell$ and γ'_ℓ to each server S_j $(j \in \mathcal{I}_{2k-1})$. Each server S_j, once it holds k n-tuples $\mathbf{\Gamma}'_\ell$ and k values γ'_ℓ $(\ell \in \mathcal{I}_{2k-1})$, calculates by interpolation (see Sect. 2.1.1.3) $\mathbf{\Gamma}' = \left[G'_1(0), G'_2(0), \ldots, G'_n(0) \right]$ and $\gamma' = H'(0)$.

4. The last operation consists, for each server S_ℓ $(\ell \in \mathcal{I}_{2k-1})$, in checking that $\mathbf{\Gamma}' = \mathbf{0}_n$ and $\gamma' = 1$. If these two conditions are satisfied, the received shares correspond to a legitimate choice of the receiver.

4.5.5 Transfer Phase Step 4 – Computation of the Requested Secret

Now, each server S_ℓ $(\ell \in \mathcal{I}_{2k-1})$ holds an n-tuple $\mathbf{\Theta}_\ell = [F_1(\ell), F_2(\ell), \ldots, F_n(\ell)]$ as its shares of the secrets of $\mathbf{\Omega}$, and a legitimate n-tuple $\mathbf{\Phi}'_\ell = \left[Z'_1(\ell), Z'_2(\ell), \ldots, Z'_n(\ell) \right]$ as its shares of the receiver's request $\mathbf{\Lambda}$.

The elements of the two n-tuples $\mathbf{\Theta}$ and $\mathbf{\Phi}'$ are polynomials belonging to $\mathbb{K}_{k-1}[X]$. It follows (see Corollary 4.6) that $\mathbf{\Theta} \cdot \mathbf{\Phi}'$ is a polynomial $\mathbb{K}_{2k-2}[X]$, that $(\mathbf{\Theta} \cdot \mathbf{\Phi}')(0) = \omega_\sigma$ and that for $\ell \in \mathcal{I}_{2k-1}$, $\lambda_\ell = \mathbf{\Theta}_\ell \cdot \mathbf{\Phi}'_\ell$ is a share of ω_σ, generated by the quasi-random polynomial $\mathbf{\Theta} \cdot \mathbf{\Phi}'$.

So, each server S_ℓ $(\ell \in \mathcal{I}_{2k-1})$ calculates the new share $\lambda_\ell = \mathbf{\Theta}_\ell \cdot \mathbf{\Phi}'_\ell$. Since at least $2k - 1$ servers participate in the computation, they are able to redistribute the resulting shares λ_ℓ, $(\ell \in \mathcal{I}_{2k-1})$, thanks to a (k, k)-threshold Shamir's scheme (using the method described in Sect. 4.4.2), to the servers $S_j (j \in \mathcal{I}_k)$.

4.5.6 Transfer Phase Step 5 – Transfer of the Requested Secret

After redistribution of the secret shares in the previous step, the set of contacted servers S_j $(j \in \mathcal{I}_k)$ collectively owns the value of $\omega_\sigma = \mathbf{\Omega}_0 \cdot \mathbf{\Phi}_0$, under the form of shares generated by a (k, k)-threshold Shamir's scheme. Each server S_j responds to the receiver's request with the value R_j, which is its share of $\mathbf{\Omega}_0 \cdot \mathbf{\Phi}_0$ generated from a sharing polynomial R of $\mathbb{K}_{k-1}[X]$. The receiver $\mathcal{R}$ interpolates the polynomial of $\mathbb{K}_{k-1}[X]$ corresponding to the k responses, and obtains ω_σ.

4.6 Security Evaluation of the Protocol

In this section it is demonstrated that the proposed protocol satisfies all desirable security conditions listed in Sect. 2.2.3 (i.e., conditions C_1, C_2 with $\lambda_R = k-1$, C_3 with $\lambda_S = k-1$, $C_{4.1}$, and $C_{4.2}$ with $\lambda_C = k-1$).

4.6.1 Correctness

Theorem 4.8. *The proposed DOT protocol, described in Figs. 4.1 and 4.2, satisfies security condition* C_1.

Proof. We demonstrate that $R(0) = \omega_\sigma$.

First, we observe that the secret input $\mathbf{\Omega}$ of the sender and the secret input $\mathbf{\Delta}$ of the receiver satisfy the relation $\mathbf{\Omega} \cdot \mathbf{\Delta} = \omega_\sigma$. In the set-up phase (see Sect. 4.5.1), the n-tuple of quasi-random polynomials $\mathbf{\Theta}$ is such that $\mathbf{\Theta}_0 = \mathbf{\Omega}$. Similarly, the n-tuple $\mathbf{\Phi}$ of quasi-random polynomials of $\mathbb{K}_{k-1}[X]$ generated in the first step of the transfer phase (see Sect. 4.5.2) is such that $\mathbf{\Phi}_0 = \mathbf{\Delta}$. In the second step of the transfer phase (see Sect. 4.5.3), the n-tuple $\mathbf{\Phi}'$ resulting from the shares redistribution is such that $\mathbf{\Phi}'_0 = \mathbf{\Phi}_0$. It follows that $\mathbf{\Theta}_0 \cdot \mathbf{\Phi}'_0 = \omega_\sigma$. The verification step (see Sect. 4.5.4) modifies neither $\mathbf{\Theta}$ nor $\mathbf{\Phi}'$. In the fourth step of the transfer phase (see Sect. 4.5.5), the servers S_ℓ ($\ell \in \mathcal{I}_{2k-1}$) collectively calculate the polynomial $Q = \mathbf{\Theta} \cdot \mathbf{\Phi}'$. Hence, $Q(0) = (\mathbf{\Theta} \cdot \mathbf{\Phi}')(0) = \mathbf{\Theta}_0 \cdot \mathbf{\Phi}'_0 = \omega_\sigma$. The redistribution at the end of the same step replaces the polynomial Q of $\mathbb{K}_{2k-2}[X]$ with a polynomial R of $\mathbb{K}_{k-1}[X]$, such that $R(0) = Q(0)$. We note that each server S_j ($j \in \mathcal{I}_k$) returns to the receiver the share $\lambda_j = R(j)$. Hence, in the last step of the protocol (see Sect. 4.5.6), the k shares λ_j are sufficient for the receiver to interpolate R and to determine ω_σ. □

Therefore, security condition C_1 is guaranteed.

4.6.2 Receiver's Privacy

Theorem 4.9. *The proposed DOT protocol, which is depicted in Figs. 4.1 and 4.2, satisfies security condition* C_2 *with* $\lambda_R = k-1$.

Proof.

The index σ chosen by the receiver is represented under the form of an n-tuple $\mathbf{\Delta} = [\, \delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_n^\sigma \,]$ of private values. The receiver's input to the protocol consists of shares of these values. That is, each element δ_i^σ, which is either zero or one, is distributed amongst the set of k servers S_ℓ ($\ell \in \mathcal{I}_k$), using a (k, k)-threshold Shamir's scheme. This is achieved by generating an n-tuple $\mathbf{\Phi} = [\, Z_1, Z_2, \ldots, Z_n \,]$ of n quasi-random polynomials of $\mathbb{K}_{k-1}[X]$, such that $Z_i(0) = \delta_i^\sigma$.

In order to breach the privacy of the receiver, a set of $k-1$ colluding servers (during the execution of the protocol, or after completion of the protocol) should be able to determine at least one of the values δ_i^σ. The set of $k-1$ collaborating servers, however, owns $k-1$ shares corresponding to each values δ_i^σ associated with a (k, k)-threshold Shamir's scheme. Due to the perfectness of Shamir's secret sharing scheme, every set of $k-1$ shares provides the coalition with absolutely no information about the relevant secret. That is, the inputs of the receiver guarantees her privacy, in the first step of the transfer phase of the protocol. Although in the next step of the protocol, the set of k servers redistribute the private values δ_i^σ, to $2k-1$ servers, this transformation does not leak any information to a set of $k-1$ servers (see Theorem 4.7). In the same way, the redistributions of $\delta_i^\sigma(\delta_i^\sigma - 1) = 0$, $\sum_{j\in\mathcal{I}_n} \delta_i^\sigma = 1$ and δ_i^σ $(1 \in \mathcal{I}_n)$ presented in Sects. 4.5.4 and 4.5.5 preserve the privacy of the receiver.

One may argue that, after completion of the protocol, the knowledge of the output ω_σ may help the set of $k-1$ servers to determine the choice of the secret. However, this is not the case, since the shares held by the $k-1$ servers after the redistribution may be considered as generated from a (k, m)-threshold Shamir's scheme, and thus $k-1$ shares of any secret provides no information about the secret. □

Consequently, the security condition C_2 with $\lambda_R = k-1$ is guaranteed.

4.6.3 Sender's Security with Respect to a Coalition of Servers and the Receiver

At the level of the servers, the role of the n-tuple $\mathbf{\Omega}$ is the same as the role of the n-tuple $\mathbf{\Phi}'$. That is, a set of $k-1$ colluding servers cannot learn any of the secrets $\omega_1, \omega_2, \ldots, \omega_n$ (see previous section). Note that, in this scenario, there is no advantage for the coalition of $k-1$ servers to collude with the receiver to breach the sender's security, since the receiver has no input to contribute in this attack. Indeed, even if the receiver initiates the protocol by contacting only $k-1$ servers (no matter what her input to the k-th selected server is), she cannot learn any of the secrets. This is because, the $k-1$ shares she receives from the servers are considered as generated by a (k, m)-threshold Shamir's scheme, and thus provide no information about the secrets.

Remark 4.10. *One may note that, contrary to the protocol presented in [19, 20], the protocol presented in this chapter does not require the use of masks. Indeed, in the scheme, a corrupt server cannot provide the receiver with a linear combination of the secrets $\omega_1, \omega_2, \ldots, \omega_n$ because it never owns such a combination.*

4.6.4 Sender's Security with Respect to a 'Greedy' Receiver

Let $\mathcal{I}_C$ be the set of indices of $k-1$ corrupt servers. A corrupt server S_ℓ ($\ell \in \mathcal{I}_C$) may be one of the servers selected by the receiver in the first step of the transfer phase protocol. In this case, $\ell \in \mathcal{I}_k$. The worst scenario is assumed, from the sender's point of view (the best scenario from the 'greedy' receiver's point of view), i.e., that all the corrupt servers are selected by the receiver in the first step of the transfer phase of the protocol ($\mathcal{I}_C \subseteq \mathcal{I}_k$). Thus, it is assumed that the receiver has access to all information of these $k-1$ corrupt servers during the execution of the protocol.

In the initialization step, each server S_c ($c \in \mathcal{I}_C$) receives from the sender an n-tuple $\mathbf{\Omega_c} = [\, F_1(c), F_2(c), \ldots, F_n(c) \,]$ of n elements, where the element $F_i(c)$ is the share allocated to S_c for the secret ω_i. Clearly, the knowledge of $k-1$ shares of any secret ω_i, provides the receiver with no useful information about the secret ω_i.

In the first step of the transfer phase of the protocol, the receiver transmits her inputs to servers S_ℓ ($\ell \in \mathcal{I}_k$). Obviously, she cannot learn any information about $\omega_1, \omega_2, \ldots, \omega_n$ from her inputs.

In the second step of the transfer phase, the contacted servers redistribute the receiver's private values δ_i^σ ($1 \le i \le n$) amongst the servers S_ℓ ($\ell \in \mathcal{I}_{2k-1}$), using the method described in Sect. 4.4.2. The receiver has access to all partial shares received by servers S_c ($c \in \mathcal{I}_C$), and has also access to all partial shares sent by servers S_c ($c \in \mathcal{I}_C$) to servers S_ℓ ($\ell \in \mathcal{I}_{2k-1}$). At the end of this step, the receiver's private values, δ_i^σ ($1 \le i \le n$), are shared amongst the servers S_ℓ ($\ell \in \mathcal{I}_{2k-1}$). The shares are considered as generated from $(k, 2k-1)$-threshold Shamir's schemes. Although the receiver, for each private value δ_i^σ, has only access to $k-1$ shares owned by servers S_c ($c \in \mathcal{I}_C$), she can figure out all the $2k-1$ shares elaborated by servers S_ℓ ($\ell \in \mathcal{I}_{2k-1}$). Indeed, for $i = 1, 2, \ldots, n$, the receiver knows δ_i^σ and $k-1$ shares of δ_i^σ; so she can reconstruct the associated sharing polynomial generated in the $(k, 2k-1)$-threshold Shamir's scheme, and from the polynomial, the corresponding shares held by servers S_ℓ ($\ell \in \mathcal{I}_{2k-1}$). That is, the receiver is able to find out all the elements of the n-tuple $\mathbf{\Phi}'_\ell$ ($\ell \in \mathcal{I}_{2k-1}$).

In the fourth step of the transfer phase, each server S_ℓ ($\ell \in \mathcal{I}_{2k-1}$) computes $\lambda_\ell = \mathbf{\Omega}_\ell \cdot \mathbf{\Phi}'_\ell$. The receiver is able to multiply the corresponding shares of two n-tuples $\mathbf{\Omega}_c$ and $\mathbf{\Phi}'_c$ ($c \in \mathcal{I}_C$), and thus learns $k-1$ shares of $\omega_i \times \delta_i^\sigma$ ($1 \le i \le n$). She also knows the value of $\omega_i \times \delta_i^\sigma$ ($1 \le i \le n$), because if $\delta_i^\sigma = 0$, then the $\omega_i \times \delta_i^\sigma = 0$, and if $\delta_i^\sigma = 1$, then $\omega_i \times \delta_i^\sigma = \omega_\sigma$, which is the secret that the receiver learns at the end of the protocol. However, contrary to the previous step, knowing a value $\omega_i \times \delta_i^\sigma$ ($1 \le i \le n$) and $k-1$ of its shares is not sufficient to learn all other shares. This is because in the fourth step of the transfer phase, the value and the shares are associated with a polynomial of $\mathbb{K}_{2k-2}[X]$ (see Sect. 4.5.5), generated in a $(2k-1, 2k-1)$-threshold scheme. Consequently, the

receiver cannot learn any useful information about ω_i ($1 \leq i \leq n, i \neq \sigma$). Note that the redistribution of secrets from a $(2k-1, 2k-1)$-threshold scheme to a (k, k)-threshold scheme is completely secure (see Sect. 4.4.2) and therefore provides the receiver with no information about $\omega_1, \omega_2, \ldots, \omega_n$.

Finally, in the last step of the transfer phase, the data held by servers S_c ($c \in \mathcal{I}_C$) are not modified; the servers S_j ($j \in \mathcal{I}_k$) transmit to the receiver k shares of ω_σ, generated by a (k, k)-threshold scheme and the receiver learns the secret ω_σ. That is, if the receiver generates a valid request in the first step of the transfer phase, there is no way for her to learn more than the requested secret. If the request is not legitimate, the protocol is aborted by the honest servers in the third step of the transfer phase. Thus, in both situations, the following is proved.

Theorem 4.11. *The proposed DOT protocol, which is depicted in Figs. 4.1 and 4.2, satisfies security condition C_4 with $\lambda_C = k - 1$.*

4.7 Conclusion

In this chapter was presented a new one-round, polynomial interpolation-based, unconditionally secure DOT protocol. The protocol significantly improves the security of DOT protocols. If the efficiency of the protocol is compared to the efficiency of the polynomial interpolation-based DOT protocol presented in [19, 20], it is observed that the protocol presented in this chapter requires more computation on the servers' sides, but less computation on the sender and receiver's sides, as shown in Table 4.1.

In contrast to the protocol described in this chapter, the DOT protocol proposed in the next chapter allows a receiver to obtain a chosen secret, even in the presence of malicious servers which may try to disrupt the protocol's execution. Some mechanisms are included in the protocol, so that the secret obtained by the receiver is the chosen secret, in spite of servers not responding or providing incorrect responses.

Table 4.1: Efficiency of Distributed Oblivious Transfer Protocols

Party	**Blundo et al.'s Protocol**	**Proposed Protocol**
Sender $\mathcal{S}$	1 distribution of n secrets $c_0\omega_0, c_1\omega_1, \ldots, c_{n-1}\omega_{n-1}$, n masks $c_0, c_1, \ldots, c_{n-1}$ and $2 \times (n-1)$ masks r_1^1, $r_2^1, \ldots, r_{n-1}^1, r_1^2, r_2^2, \ldots, r_{n-1}^2$	1 distribution of n secrets $\omega_1, \omega_2, \ldots, \omega_n$
Receiver $\mathcal{R}$	1 distribution of n private values (0 or 1), 1 polynomial interpolation	1 distribution of n private values (0 or 1), 1 polynomial interpolation
Server S_ℓ (worst case)	1 generation of 2 shares from degree 1 polynomials	n transformations from a (k, k)- to a $(k, 2k-1)$-threshold scheme, $(n+1)$ transformations from a $(2k-1, 2k-1)$- to a $(2k-1, 2k-1)$-threshold scheme, 1 transformation from a $(2k-1, 2k-1)$- to a (k, k)-threshold scheme, 1 generation of the dot product of two n-tuples, 1 polynomial interpolation

5

A Verifiable Distributed Oblivious Transfer Protocol

In the various *distributed oblivious transfer* protocols 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, either on the sender's secrets, or 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 so 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.

In this chapter is studied the first unconditionally secure distributed oblivious transfer protocol in the presence of an active adversary who may corrupt up to $k-1$ servers. In addition to the active adversary, it is also assumed that the sender may corrupt up to $k-1$ servers to learn the choice of the receiver. Similarly, the receiver may corrupt up to $k-1$ servers to learn more than the chosen secret. However, it is assumed that the sender, the receiver, and the active adversary do not collaborate with each other. The distributed oblivious transfer protocol allows the receiver to contact $4k-3$ servers to obtain one secret, while the required security is maintained.

5.1 Introduction

In the unconditionally secure *distributed oblivious transfer* (DOT) schemes presented in [8, 19, 20, 74, 78], 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 to reconstruct the chosen secret. The security of these protocols may be assessed thanks to the model defined in Sect. 2.2.3.

In particular, the *correctness* property (security condition C_1) guarantees that the receiver obtains the secret she has chosen. However, the security model of the protocols mentioned above is semi-honest, with regards to the servers. That is, no mechanism prevents a set of up to $k-1$ malicious servers from disrupting a protocol such that the secret obtained by the receiver does not correspond to the chosen secret. Such a mechanism could allow (*a*) the servers to check that the requests sent by the receiver are consistent and (*b*) the receiver to collect redundant shares to detect and discard inconsistent shares returned by the malicious servers.

This kind of mechanism is introduced in one of the polynomial interpolation-based protocols presented by Blundo, D'Arco, De Santis, and Stinson [19, 20]. In this context, security 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 up to λ_M malicious servers.

The resulting protocol guarantees security conditions C_1' with $\lambda_M = k-1$, C_2 with $\lambda_R = k-1$, C_3 with $\lambda_S = k-1$ and $C_{4.1}$, despite the participation of malicious servers to the protocol. Blundo et al. have proved that security condition $C_{4.2}$ cannot be satisfied, for any $\lambda_C > 0$, 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 chapter is organized as follows. The next section shortly describes Blundo et al.'s protocol [20] and investigates verifiable secret sharing schemes. In Sect. 5.3 a few notations and definitions used in the rest of the chapter are introduced. Then, in Sect. 5.4, the model related to the DOT protocol presented in the chapter is described and an overview the protocol is given. The main components of the protocol are described in Sect. 5.5 and the

protocol itself is presented in Sect. 5.6. In Sect. 5.7, the security of the protocol is analysed and a conclusion is given in Sect. 5.8.

5.2 Related Works

Although there have been a few DOT protocols studied in the past 15 years, e.g. [19, 20, 74, 78, 101, 102], the verifiability of distributed shares in such protocols was rarely tackled. The main contribution to the subject was made by Zhong and Yang [101, 102], but the setting of their proposed scheme is conditionally secure (difficulty to compute a discrete logarithm).

The first unconditionally secure verifiable DOT protocol is presented. The protocol is adapted from the main component of the DOT protocol introduced by Blundo et al. [19, 20]. This component is described first.

5.2.1 Blundo et al.'s Distributed Oblivious Transfer Protocol

The basic principles underlying DOT protocols are conceptually similar. In the original DOT protocol [74] introduced by Naor and Pinkas, as well as in its generalization [19, 20] 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 [20] is described in Sect. 2.3.2 and the full description in Sect. 2.3.3.

5.2.2 Verifiable Secret Sharing Schemes

A component common to all unconditionally secure DOT protocols is the threshold secret sharing scheme introduced by Shamir [89] in 1979 (see an overview of the protocol in Sect. 2.3.1). In this scheme, a 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 players to reconstruct the secret are genuine. In 1985, Chor, Goldwasser, Micali, and Awerbuch [30] 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 [96] 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 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 VSS schemes were introduced by Ben-Or, Goldwasser, and Wigderson [14] and by Chaum, Crépeau, and Damgård [27]. However, the correctness of the VSS scheme presented in [27] is not guaranteed: the reconstructed secret may be incorrect with a very small probability. A similar weakness can be found in [83, 84] and [34].

The unconditionally secure VSS scheme introduced by Ben-Or et al. [14] 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 [35] and a simpler version was proposed by Desmedt, Kurosawa, and Van Le [43]. However, in the original scheme and in 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 [95], Stinson and Wei introduced an unconditionally secure VSS scheme 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 scheme [53] similar to Stinson and Wei's scheme, tolerating also up to $t = \lfloor \frac{n}{4} \rfloor$ cheaters and with no share publicly disclosed. D'Arco and Stinson [40] improved this VSS scheme in 2002 using a feature introduced in [53] to detect inconsistencies.

However, in [95] and in [40], 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 schemes is not polynomial. Inversely, in [53], 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 [45]. For these reasons, it was chosen to replace the secret sharing scheme component used in [20] with Gennaro et al.'s VSS scheme.

5.3 Preliminaries

Throughout this chapter, operations are executed in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power). The number of secrets involved in the protocol is n ($n > 1$) and the number of servers is m ($m > 1$). It is assumed that $p \geq \max(n, m)$. In addition, by an abuse of language, a polynomial and its corresponding polynomial function will not be differentiated.

The definition of quasi-random polynomial (see Sect. 2.1) is extended to bivariate polynomials.

Definition 5.1. *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* $\mathbb{K}$*, a polynomial* $F = \sum_{i=0}^{k}\sum_{j=0}^{k} f_{i,j}X^iY^j$ *of* $\mathbb{K}_k[X,Y]$ *is said to be* 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 5.2. *In a* (k, m)*-threshold secret sharing scheme involving a dealer* $\mathcal{D}$*,* m *players* $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$ *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* $\mathcal{P}_1$*,* $\mathcal{P}_2, \ldots, \mathcal{P}_m$*, then the* m *shares are said to be* k-consistent.

Definition 5.3. *In a* (k, m)*-threshold secret sharing scheme involving a dealer* $\mathcal{D}$*,* m *players* $\mathcal{P}_1$*,* $\mathcal{P}_2, \ldots, \mathcal{P}_m$ *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 to be* k-resilient.

Notations:
If $\boldsymbol{v} = [\,f_1, f_2, \ldots, f_n\,]$ is an n-tuple of polynomials and i an element of $\mathbb{K}$, the n-tuple $[\,f_1(i), f_2(i), \ldots, f_n(i)\,]$ is denoted by $\boldsymbol{v}(i)$.

5.4 Model

The setting of protocol presented in this chapter encompasses a sender $\mathcal{S}$ who holds n secrets $\omega_0, \omega_1, \ldots, \omega_{n-1}$ ($n > 1$), elements of the field $\mathbb{K}$, a receiver $\mathcal{R}$ who wishes to obtain a secret ω_σ ($\sigma \in \mathcal{I}_\mathrm{n} = [\![\,0, n-1\,]\!]$), and m servers S_i ($m > 1$ and $i \in \mathcal{I}_\mathrm{m} \subseteq \mathbb{K}^*$, where $|\mathcal{I}_\mathrm{m}| = m$). Without loss of generality, it is assumed that $\mathcal{I}_\mathrm{m} = \{\,1, 2, \ldots, m\,\}$. In addition to these parties, it is necessary to take into account an adversary whose characteristics are defined below.

Like other DOT protocols, the protocol is composed of two phases: a *set-up phase* and a *transfer phase*. During the set-up phase, 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 transfer phase, the receiver has to contact $c \geq 4k-3$ servers ($c \leq m$) to collect enough shares to construct ω_σ.

5.4.1 Adversary Model

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

- The sender $\mathcal{S}$, with possibly a coalition of 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 corrupt 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 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.

It is assumed 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, it is assumed 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 chapter, only static parties are considered; 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, during the execution of the protocol, some servers may be disqualified. It is assumed that a mechanism prevents the disqualified servers from keeping on participating in the protocol.

5.4.2 Communication Model

The protocol requires the availability of private communication channels between the sender and the servers, the receiver and the servers and amongst the servers. It also requires

a broadcast channel, allowing all participants to receive simultaneously information sent by one participant through this channel.

It is assumed 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 (see Sect. 2.2.1).

5.4.3 Principle of the Protocol

The protocol introduced by Blundo et al. (see Sect. 5.2.1) 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. 5.1), 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 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_s^σ $(1 \leq s \leq n-1)$ distributes shares generated by Gennaro et al.'s VSS scheme to these servers. For each secret input, the VSS scheme 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 request transmitted by the receiver, consisting of $n - 1$ shares. 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.

5.5 Components of the Protocol

The protocol is mainly based on two components.

The first one is an error-correcting codes decoding scheme [52, 85]. 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 scheme introduced by Gennaro et al. [53]. 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.

5.5.1 Error-Correcting Codes Decoding Scheme

Let $\mathcal{D}$ be an honest dealer holding a secret s, $\mathcal{P}_i$ $(i = 1, 2, \ldots, m)$ a player and C a combiner wishing to obtain s. It is assumed that $\mathcal{D}$ uses for example a (k, m)-threshold Shamir's secret sharing 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 C collects k shares from the players, she can reconstruct s, but if C collects $k-1$ or fewer shares, she does not obtain any information on s. It is also assumed that up to t players may cheat; a cheating player $\mathcal{P}_c$, when requested by C to provide the share s_c received from $\mathcal{D}$, may transmit to C a value $s'_c \neq s_c$.

If C contacts k players, she will collect k shares (correct or incorrect) and will be able to interpolate a polynomial f' of $\mathbb{K}_{k-1}[X]$ and to calculate $s' = f'(0)$. If only one share is incorrect, $s' \neq s$. So, C is considered to be allowed to contact ℓ players $(k \leq \ell \leq m)$ to collect some redundant information.

If the ℓ shares are not k-consistent, a protocol is proposed hereafter so that if $\ell \geq k+2t$, then 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 [85] 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 element $\alpha_i \in \mathbb{K}$. The values α_1, $\alpha_2, \ldots, \alpha_m$ 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 $\boldsymbol{a} = [\, a_0, a_1, \ldots, a_{k-1} \,]$ encoded as the codeword $\boldsymbol{D} = [\, D_1, D_2, \ldots, D_m \,]$, 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 C collects ℓ shares ($k \leq \ell \leq m$), without loss of generality from players $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell$, and checks that the collected shares are k-consistent. To perform this task, 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 $\mathbb{K}_{k-1}[X]$. For all remaining shares $[\alpha_j, D_j]$ ($k+1 \leq j \leq \ell$), C checks that $f(\alpha_j) = D_j$. If the $\ell - k$ equalities are satisfied, the shares are k-consistent and C calculates $s = f(0)$.

If incorrect shares were detected, they could be identified and corrected thanks to algorithms like the Berlekamp-Massey algorithm [15] introduced by Berlekamp. However, if the main objective of error-correcting codes is to restore original codes from corrupted codes, the goal here 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 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 [52] is preferred. This algorithm allows C to reconstruct f, in spite of up to $t \leq \frac{\ell-k}{2}$ incorrect shares.

First, the combiner C generates two polynomials g_0 and g_1 such that:

- $g_0(x) = \prod_{i=1}^{\ell} (x - \alpha_i)$ and
- g_1 is interpolated from the received shares $D_1, D_2, \ldots, D_\ell$. 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, C calculates a partial greatest common divisor polynomial g between g_0 and g_1. This polynomial g is determined using the extended Euclidean algorithm. More precisely, 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, 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, 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 $s = f(0)$.

5.5.2 Verifiable Secret Sharing Scheme

The setting of Gennaro et al.'s VSS scheme [53] encompasses a dealer $\mathcal{D}$ who holds a secret ω, m players $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$, each of them receiving a (k, m)-threshold share of ω and a combiner C collecting the shares from ℓ ($1 < \ell < m$) players $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell$. 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 C. The protocol guarantees

that, if $\ell \geq 4t + 1$, 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 as follows.

First, $\mathcal{D}$ generates a quasi-random bivariate polynomial F in $\mathbb{K}_{k-1}[X, Y]$. The polynomial function $F(x, i)$ where $i \in [\![1, m]\!]$ is denoted by $f_i(x)$. Similarly, the polynomial function $F(i, y)$ where $i \in [\![1, m]\!]$ is denoted by $g_i(y)$. 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 $\mathcal{P}_j$ $(1 \leq j \leq m)$ an element $r_{i,j} \in_R \mathbb{K}$ 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 $\{ L_1, L_2, \ldots, L_m \}$ 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 $L_1, L_2, \ldots, L_m$ into a graph $\mathcal{G} = [V, E]$ where $V = \{ \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m \}$ and $\left[\mathcal{P}_i, \mathcal{P}_j \right] \in E$ 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 5.4. *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 App. E.3, $\mathcal{P}_i$ 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 $\left[\mathcal{P}_i, \mathcal{P}_j \right] \in M_{\text{al}}$ and the set $H = \{ \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m \} \setminus \overline{H}$. 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. 5.5.1 allows C to determine ω from the shares collected from the players of $H \cup D$.

Like Shamir's secret sharing scheme, Gennaro et al.'s VSS scheme is perfect, i.e., the knowledge of $k - 1$ or fewer shares $[f_i, g_i]$ leaves ω completely undetermined.

Let $\mathcal{I}_{\mathrm{m}} = \{ 1, 2, \ldots, m \}$ $(m > 1)$ be a subset of $\mathbb{K}^*$ and S_j be a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Input The sender $\mathcal{S}$, contributes with n secrets $\omega_0, \omega_1, \ldots, \omega_{n-1}$ of $\mathbb{K}$
The receiver $\mathcal{R}$, chooses an index $\sigma \in \mathcal{I}_{\mathrm{n}} = [\![0, n-1]\!]$, and contributes with $n-1$ private values $\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma \in \{ 0, 1 \}$

Output $\mathcal{R}$ is either detected as a cheater and 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 function $Q(x, y_1, y_2, \ldots, y_{n-1}) = P(x) + \sum_{i=1}^{n-1} (\omega_i - \omega_0) y_i$ 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 $(\ell \in \mathcal{I}_{\mathrm{m}})$ the $(n-1)$-variate polynomial function $F_\ell(y_1, y_2, \ldots, y_{n-1}) = Q(\ell, y_1, y_2, \ldots, y_{n-1})$.

Figure 5.1: A Verifiable $(4k-3, m)$-DOT-$\binom{n}{1}$ (Set-up Phase)

5.6 Description of the Protocol

This section presents a verifiable DOT protocol (see Figs. 5.1 and 5.2). The protocol is composed of five phases, described hereafter.

5.6.1 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

$$Q(x, y_1, y_2, \ldots, y_{n-1}) = P(x) + \sum_{j=1}^{n-1} (\omega_j - \omega_0) y_j \,.$$

For each index $\ell \in \mathcal{I}_{\mathrm{m}}$, $\mathcal{S}$ builds an $(n-1)$-variate polynomials $F_\ell = Q(\ell, y_1, y_2, \ldots, y_{n-1})$ 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.

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

1. $\mathcal{R}$ chooses $\sigma \in \mathcal{I}_{\mathrm{n}}$ and $\mathcal{I} \subseteq \mathcal{I}_{\mathrm{m}}$, a set of $c \geq 4k-3$ servers to contact.
2. $\mathcal{R}$ generates an $(n-1)$-tuple $\boldsymbol{\Theta} = [\, G_1, G_2, \ldots, G_{n-1} \,]$ 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_s^{σ}. The polynomial function $G_s(x, i)$ $(i \in \mathcal{I})$ is denoted by $f_{s,i}(x)$ and the polynomial function $G_s(i, y)$ $(i \in \mathcal{I})$ is denoted by $g_{s,i}(y)$.
3. $\mathcal{R}$ generates c tuples $\boldsymbol{V}_i = \left[f_{1,i}, f_{2,i}, \ldots, f_{n-1,i}) \right]$ of polynomials $(i \in \mathcal{I})$. Similarly, $\mathcal{R}$ generates c tuples $\boldsymbol{W}_i = \left[g_{1,i}, g_{2,i}, \ldots, g_{n-1,i} \right]$ of polynomials $(i \in \mathcal{I})$.
4. $\mathcal{R}$ transmits to S_i $(i \in \mathcal{I})$ the pair of tuples $[\, \boldsymbol{V}_i, \boldsymbol{W}_i \,]$.
5. For $s = 1, 2, \ldots, n-1$, S_i $(i \in \mathcal{I})$ sends to the server S_j $(j \in \mathcal{I})$ a random element $r_{s,i,j} \in_R \mathbb{K}$.

Phase 3 **– Detection of Cheaters**

1. S_i $(i \in \mathcal{I})$ broadcasts $f_{s,i}(j) + r_{s,i,j}$ and $g_{s,i}(j) + r_{s,j,i}$, for $s = 1, 2, \ldots, n-1$. From the broadcast values and for $s = 1, 2, \ldots, n-1$, S_i builds three sets of servers H_s, D_s and C_s, following the technique described in Sect. 5.5.2. If $|C_s| > k-1$, then $\mathcal{R}$ has cheated and the protocol stops.
2. S_i $(i \in \mathcal{I})$ 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 by $\mathcal{I}_{\mathcal{H}}$.
3. S_i $(i \in \mathcal{I}_{\mathcal{H}})$ calculates $\boldsymbol{\Phi}_i = [\, Z_1(i), Z_2(i), \ldots, Z_{n-1}(i) \,]$. 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. 5.5.1.

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 $\mathbb{K}_{k-1}[X]$, thanks to the error-correcting codes decoding scheme described in Sect. 5.5.1 and calculates $\omega_{\sigma} = R(0)$.

Figure 5.2: A Verifiable $(4k-3, m)$-DOT-$\binom{n}{1}$ (Transfer Phase)

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

The receiver $\mathcal{R}$ chooses the index $\sigma \in \mathcal{I}_{\mathrm{n}}$ of the secret ω_σ she wishes to obtain as well as a set $\mathcal{I} \subseteq \mathcal{I}_{\mathrm{m}}$ of $c \geq 4k-3$ indices of servers. The index σ is coded under the form of a tuple $[\,\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma\,]$ of $n-1$ elements of $\{\,0,1\,\}$.

Then, for each secret input δ_s^σ, $\mathcal{R}$ generates and distributes shares of δ_s^σ according to the VSS scheme sharing phase of the protocol described in Sect. 5.5.2.

That is, $\mathcal{R}$ generates for $s = 1, 2, \ldots, n-1$, a quasi-random bivariate polynomial $G_s \in \mathbb{K}_{k-1}[X,Y]$ such that $G_s(0,0) = \delta_s^\sigma$. For $i \in \mathcal{I}$, the polynomial function $G_s(x,i)$ is denoted by $f_{s,i}(x)$ and the polynomial function $G_s(i,y)$ by $g_{s,i}(y)$. The polynomial function Z_s by $Z_s(y) = G_s(0,y)$ is also defined, for $s = 1, 2, \ldots, n-1$. The polynomial $Z_s \in \mathbb{K}_{k-1}[X]$ satisfies the c equalities $Z_s(i) = f_{s,i}(0)$ $(i \in \mathcal{I})$. The receiver $\mathcal{R}$ distributes to server S_i $(i \in \mathcal{I})$ the two $(n-1)$-tuples of polynomials $\boldsymbol{V}_i = \left[\, f_{1,i}, f_{2,i}, \ldots, f_{n-1,i} \,\right]$ and $\boldsymbol{W}_i = \left[\, g_{1,i}, g_{2,i}, \ldots, g_{n-1,i} \,\right]$.

Then, for $s = 1, 2, \ldots, n-1$, each contacted server S_i $(i \in \mathcal{I})$ sends to the server S_j $(j \in \mathcal{I})$ a random element $r_{s,i,j} \in_R \mathbb{K}$.

5.6.3 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.

5.6.3.1 Categorization of Servers

Each server S_i $(i \in \mathcal{I})$ broadcasts $f_{s,i}(j) + r_{s,i,j}$ and $g_{s,i}(j) + r_{s,j,i}$, for $s = 1, 2, \ldots, n-1$. From the broadcast values, each server builds for each value $s = 1, 2, \ldots, n-1$, c lists $L_{s,1}, L_{s,2}, \ldots, L_{s,c}$ 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 $L_{s,1}, L_{s,2}, \ldots, L_{s,c}$ into a graph $\mathcal{G}_s = [\,V_s, E_s\,]$ where $V_s = \{\, S_{i_1}, S_{i_2}, \ldots, S_{i_c} \,\}$ $(i_1, i_2, \ldots, i_c \in \mathcal{I})$ and $\left[\, S_i, S_j \,\right] \in E_s$ if $S_i \notin L_{s,j}$ and $S_j \notin L_{s,i}$. The next step consists for each server S_i $(i \in \mathcal{I})$ in building for $s = 1, 2, \ldots, n-1$ 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 App. E.3, S_i $(i \in \mathcal{I})$ determines, for each value s $(s = 1, 2, \ldots, n-1)$ 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 $\left[\, S_i, S_j \,\right] \in M_s$ 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.

5.6.3.2 Overall Verification

Then, each server S_i ($i \in \mathcal{I}$) 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 as 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 by $\mathcal{I}_{\mathcal{H}}$. A server S_d belongs to the set $\mathcal{D}$ of disqualified servers if $d \in \mathcal{I} \setminus \mathcal{I}_{\mathcal{H}}$. The servers in $\mathcal{D}$ do not participate to the rest of the protocol.

5.6.3.3 Recovering of Shares

Each server S_i ($i \in \mathcal{I}_{\mathcal{H}}$) calculates the $(n-1)$-tuple

$$\mathbf{\Phi}_i = [\, Z_1(i), Z_2(i), \ldots, Z_{n-1}(i) \,] \ .$$

The share $Z_s(i)$ ($1 \leq s \leq n-1$) 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. 5.5.1.

5.6.4 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 an $(n-1)$-tuple $\mathbf{\Phi}_i = [\, Z_1(i), Z_2(i), \ldots, Z_{n-1}(i) \,]$ of secret inputs shares. Therefore, S_i is able to calculate the value $\mu_i = F_i(\mathbf{\Phi}_i)$.

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

5.6.5 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. 5.5.1 to interpolate a polynomial R of $\mathbb{K}_{k-1}[X]$. Assuming $\mathcal{R}$ distributed shares of $\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma$ in Phase 2, $\mathcal{R}$ can easily calculate $\omega_\sigma = R(0)$.

5.7 Security of the Protocol

In this section it is shown (proof sketch) that the proposed protocol satisfies all desirable conditions C_1' with $\lambda_{\mathrm{M}} = k-1$, C_2 with $\lambda_{\mathrm{R}} = k-1$, C_3 with $\lambda_{\mathrm{S}} = k-1$ and $C_{4.1}$.

5.7.1 Correctness

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

Each of the $n-1$ sets $H_s \cup D_s$ ($1 \leq s \leq n-1$) determined in the first step of Phase 3 contains at least $c-(k-1)$ servers holding k-consistent shares. If $t = k-1$, the number of servers in $H_s \cup D_s$ is at least $k+2t$, including up to t malicious servers, because $c \geq 4k-3$. 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. 5.5.1 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 $|C| \leq k-1$, it cannot be proved that $\mathcal{R}$ cheated. It follows that $|\mathcal{H}| = c - |C| \geq c-(k-1)$. Again, the parameter t is defined by $t = k-1$. Because $c \geq 4k-3$, it holds $|\mathcal{H}| \geq k+2t$.

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 an $(n-1)$-tuple $\mathbf{\Phi_i} = [\, Z_1(i), Z_2(i), \ldots, Z_{n-1}(i) \,]$ 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. 5.5.1 and calculate the chosen secret $\omega_\sigma = R(0) = Q(0, \delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma)$.

To conclude, condition C_1' with $\lambda_{\mathrm{M}} = k-1$ is satisfied.

5.7.2 Receiver's Privacy

To demonstrate that the receiver's privacy is guaranteed, it is sufficient to show that the sender, but also the sender and $k-1$ corrupt servers, cannot obtain any information on the receiver's choice, during or after the execution of the protocol.

The index σ chosen by the receiver is represented under the form of an $(n-1)$-tuple $[\,\delta_1^\sigma, \delta_2^\sigma, \ldots, \delta_{n-1}^\sigma\,]$ of private values. The receiver's input to the protocol consists of shares of these values. That is, each element δ_s^σ, which is either zero or one, is distributed amongst the c selected servers S_i ($i \in \mathcal{I}$), using Gennaro et al.'s VSS scheme. This is achieved by generating a tuple $[\,G_1, G_2, \ldots, G_{n-1}\,]$ of $n-1$ quasi-random bivariate polynomials of $\mathbb{K}_{k-1}[X,Y]$, such that $G_s(0,0) = \delta_s^\sigma$.

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

For each private value δ_s^σ, a server S_i ($i \in \mathcal{I}$) receives two polynomial functions $f_{s,i} = G_s(x,i)$ 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 ($j \in \mathcal{I}$). Indeed, S_i is able to calculate $g_{s,i}(j) = f_{s,j}(i)$.

In order to breach the privacy of the receiver during the execution of the protocol, or after completion of the protocol, a set of $k-1$ colluding servers S_ℓ ($\ell \in \mathcal{I}_\mathrm{b} \subseteq \mathcal{I}$) should be able to determine at least one of the values δ_s^σ. The set of $k-1$ collaborating servers, however, holds for each value δ_s^σ, $k-1$ shares associated with a Gennaro et al.'s scheme. More precisely, the coalition holds $k-1$ shares $Z_s(\ell)$ ($\ell \in \mathcal{I}_\mathrm{b}$) and also $k-1$ shares of each sharing polynomial $f_{s,j}$ of $\mathbb{K}_{k-1}[X]$ ($j \in \mathcal{I}$). The knowledge of a k-th share of $f_{s,j}$ would allow the coalition to determine $Z_s(j)$ and by interpolation Z_s, and then $\delta_s^\sigma = Z_s(0)$. 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 with $\lambda_\mathrm{R} = k-1$ is guaranteed.

5.7.3 Sender's Security

First, it is shown that during or after the execution of the protocol, the receiver cannot obtain more than one secret and second, it is demonstrated that a coalition of $k-1$ servers with the receiver is unable to obtain information on the secrets $\omega_0, \omega_1, \ldots, \omega_{n-1}$.

5.7.3.1 Sender's Security with Respect to the Receiver

In the original protocol, if the set of indices of the contacted servers is denoted by $\mathcal{I}_k = \{ x_1, x_2, \ldots, x_k \}$, 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):

$$\begin{cases} \sum_{j=1}^{k-1} a_j x_1^j + \omega_0 (1 - \sum_{j=1}^{n-1} Z_j(x_1)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_1) = \mu_{x_1} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{j=1}^{k-1} a_j x_k^j + \omega_0 (1 - \sum_{j=1}^{n-1} Z_j(x_k)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_k) = \mu_{x_k} \end{cases} \tag{5.1}$$

With the protocol presented in this chapter, $\mathcal{R}$ collects additional shares. The $c \geq 4k-3$ collected shares are consistent, i.e., assuming $\mathcal{I} = \mathcal{I}_k \cup \{ x_{k+1}, x_{k+2}, \ldots, x_c \}$, $\mathcal{R}$ is able to built the linear system

$$\begin{cases} \sum_{j=1}^{k-1} a_j x_1^j + \omega_0 (1 - \sum_{j=1}^{n-1} Z_j(x_1)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_1) = \mu_{x_1} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{j=1}^{k-1} a_j x_k^j + \omega_0 (1 - \sum_{j=1}^{n-1} Z_j(x_k)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_k) = \mu_{x_k} \\ \sum_{j=1}^{k-1} a_j x_{k+1}^j + \omega_0 (1 - \sum_{j=1}^{n-1} Z_j(x_{k+1})) + \sum_{j=1}^{n-1} \omega_j Z_j(x_{k+1}) = \mu_{x_{k+1}} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{j=1}^{k-1} a_j x_c^j + \omega_0 (1 - \sum_{j=1}^{n-1} Z_j(x_c)) + \sum_{j=1}^{n-1} \omega_j Z_j(x_c) = \mu_{x_c} \end{cases} \tag{5.2}$$

According to Theorem E.1 (see App. E.1), the two systems (5.1) and (5.2) are identical. It follows that $\mathcal{R}$ can learn a linear combination of secrets, as in the original protocol, 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 [74] may be applied; in Phase 1, $\mathcal{S}$ randomly selects n masks $c_0, c_1, \ldots, c_{n-1} \in \mathbb{K}^*$. 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 $\mathbf{\Phi}_i$, each server S_i ($i \in \mathcal{I}_{\mathcal{H}}$) returns two shares: $P_1(i, \mathbf{\Phi}_i)$ and $P_2(i, \mathbf{\Phi}_i)$. From the collected shares, $\mathcal{R}$ is able to apply the error-correcting codes decoding scheme described in Sect. 5.5.1 and calculate the chosen masked secret and its corresponding mask.

Consequently, with the additional technique introduced by Naor and Pinkas, $\mathcal{R}$ cannot obtain more than one secret. Therefore, security condition $C_{4.1}$ is satisfied.

5.7.3.2 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 protocol and the protocol presented in this chapter 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 the protocol presented in this chapter as for the original protocol, i.e., C_3 with $\lambda_S = k-1$ is satisfied.

5.8 Conclusion

In this chapter, a variant of DOT protocol was introduced, allowing a receiver to obtain a chosen secret, even in the presence of malicious servers (see Sect. 2.2.2). These servers seek to disrupt the protocol's execution, either by not responding or by providing incorrect responses, so that the secret obtained by the receiver is different from the chosen secret.

From a performance point of view, note that Blundo et al.'s DOT protocol [20] is more efficient than the protocol presented here. This is simply because the components used in Blundo et al.'s protocol (Lagrange interpolation technique — see Sect. 2.1.1.3 — and Shamir's secret sharing scheme [89]) are more efficient than the underlying components of the protocol presented in this chapter (error-correcting codes decoding scheme [52, 85] and VSS scheme [53]). However, a performance comparison is of limited relevance since the security models of the protocols are different.

The protocol described in the next chapter is a t-out-of-n DOT protocol, allowing a receiver to obtain t secrets in one round only. The reason for designing such a protocol is that if a receiver wishes to obtain t secrets with the current one-round DOT protocols, including the verifiable DOT protocol described in this chapter, she needs to execute the protocols t times. Of course, this is inefficient.

6

A t-out-of-n Distributed Oblivious Transfer Protocol

The original unconditionally secure *distributed oblivious transfer* protocol introduced by Naor and Pinkas [74] allows a receiver to contact k servers and obtain one out of two secrets held by a sender. In its generalized version presented by Blundo, D'Arco, De Santis, and Stinson [19, 20], a receiver can choose one out of n secrets.

In this chapter is introduced the first unconditionally secure distributed oblivious transfer protocol which allows a receiver to obtain t out of n secrets in one round only, provided this receiver communicates with $k+t-1$ servers. Like other unconditionally secure polynomial-based oblivious transfer protocols, this protocol guarantees the security of the sender and the privacy of the receiver. Furthermore, the sender's security is guaranteed even if the receiver corrupts up to $k-1$ servers and, symmetrically, the receiver's privacy is guaranteed even if the sender corrupts up to $k-1$ servers. The proposed protocol is significantly more efficient than t runs of Blundo et al.'s 1-out-of-n distributed oblivious transfer protocol.

6.1 Introduction

An *oblivious transfer* (OT) protocol allows two parties — the *sender* and the *receiver* — to exchange, in total privacy, one or more secrets. To increase the availability of the secrets, the sender distributes them to m servers. This is the case in the original *distributed oblivious transfer* (DOT) protocol [74] introduced by Naor and Pinkas, as well as in its generalization [19, 20] presented by Blundo, D'Arco, De Santis, and Stinson; in these

protocols, 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.

Basically, these protocols are composed of two phases: (*i*) the *set-up phase* and (*ii*) the *transfer phase*. During the set-up phase, the sender generates pieces of his secrets — the *shares* — and sends to each of the m servers a different set of shares. In the transfer phase, the receiver chooses the index of one secret and selects k servers she intends to contact. Then, she generates shares from the chosen index and sends to each of the k selected servers a value determined by the calculated shares and by the identifier of the server. Each contacted server generates a response based on its program and the value sent by the receiver. The response is sent back to the receiver. After receiving k responses, the receiver is able to determine the chosen secret. (A detailed overview of the DOT protocol presented in [20] is described in Sect. 2.3.3.)

It is observed that if the receiver wishes to obtain $t > 1$ secrets, these protocols have to be executed t times, involving computation and communication overheads. Note that executing the set-up phase once and the transfer phase t times is not possible without compromising the security of the sender. Bellare and Micali [11] introduced an OT protocol allowing a receiver to obtain 2 out of the 3 secrets held by a sender. This protocol may be generalized so that the receiver obtains exactly $n-1$ out of the n secrets held by the sender. However, the protocol, as well as similar OT protocols (e.g., [73]), have been studied in computationally secure settings, not in unconditionally secure settings.

In this chapter is introduced the first unconditionally secure DOT protocol which allows a receiver to contact $K = k + t - 1$ servers and obtain t out of n secrets ($1 \leq t < n$) in one round only. As in [19, 20, 74], the protocol guarantees the security of the sender and the privacy of the receiver. Furthermore, the sender's security is guaranteed even if the receiver corrupts up to $k-1$ servers and, symmetrically, the receiver's privacy is guaranteed even if the sender corrupts up to $k-1$ servers. The proposed protocol is significantly more efficient than t runs of the (k, m)-DOT-$\binom{n}{1}$ presented in [19, 20]. (The notation (a, b)-DOT-$\binom{d}{c}$ corresponds to a DOT protocol where the sender holds d secrets, the receiver obtains c secrets, b servers receive shares of the sender's secrets and the receiver needs to contact a out of the b servers to obtain the chosen c secrets.)

This chapter is organized as follows: in Sect. 6.2 is introduced the security model as well as the underlying principles of the unconditionally secure (K, m)-DOT-$\binom{n}{t}$ presented in the chapter. In Sect. 6.3, a brief presentation of the protocol is given. Section 6.4 is devoted to the detailed description of the protocol and Sect. 6.5 to its security analysis. The protocol performance is discussed in Sect. 6.6, before the conclusion in Sect. 6.7.

6.2 Principle of the Protocol

The objective is to propose an unconditionally secure protocol with the same level of security as in the polynomial interpolation-based DOT protocol introduced by Blundo et al. [19, 20]. As shown by Blundo et al., their protocol satisfies security conditions C_1, C_2 with $\lambda_R = k - 1$, C_3 with $\lambda_S = k - 1$ and $C_{4.1}$ (security conditions and parameters are defined in Sect. 2.2.3). However, the protocol does not satisfy security condition $C_{4.2}$ (the protocol is insecure as soon as $\lambda_C > 0$). It is shown in Sect. 6.5 that the protocol satisfies the same security conditions, extended to t secrets. Thus, the security conditions C_1, C_2, C_3, and C_4 ($C_{4.1}$ and $C_{4.2}$) described in Sect. 2.2.3 are redefined as shown below.

C'_1. *Correctness* – The receiver is able to determine the t chosen secrets once she receives information from the contacted servers.

C'_2. *Receiver's privacy* – A coalition of up to λ'_R servers 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 with the receiver does not obtain any information about the sender's 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 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 t transcripts of interactions with the contacted servers,

- $C'_{4.1}$. 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 ($\lambda'_C > 0$) dishonest servers and the receiver does not obtain any information about secrets which were not chosen by the receiver.

In the following sections, it is shown that if the (k, m)-DOT-$\binom{n}{1}$ presented in [19, 20] is modified so that the receiver collects $K = k + t - 1$ shares instead of k shares, she obtains exactly t secrets.

The key idea underlying the t-out-of-n DOT protocol presented here is that from $K = k + t - 1$ collected shares ($1 < K \leq m$), a receiver is able to build t univariate polynomials of degree at most $k - 1$ agreeing with k shares. These t polynomials allow the receiver, who wishes to obtain t secrets, to build a free linear system of t equations in t unknowns. Solving this system results in obtaining the t chosen secrets.

6.3 Overview of the Protocol

The setting of the protocol is similar to the setting of DOT protocols in [19, 20, 74], i.e., it encompasses a sender $\mathcal{S}$ who owns n secrets $\omega_1, \omega_2, \ldots, \omega_n$ ($n > 1$) selected in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power), m servers S_i ($m > 1$ and $i \in \mathcal{I}_\mathrm{m} \subseteq \mathbb{K}^*$, where $|\mathcal{I}_\mathrm{m}| = m$) and a receiver $\mathcal{R}$. In the model considered here, the receiver wishes to obtain t secrets $\omega_{\sigma_1}, \omega_{\sigma_2}, \ldots, \omega_{\sigma_t}$ amongst the n secrets held by $\mathcal{S}$.

The protocol requires the availability of private unicast communication channels between the sender and the servers and between the receiver and the servers. Moreover, it is assumed 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 (see Sect. 2.2.1).

The protocol is composed of two phases: a *set-up phase* and a *transfer phase*. During the set-up phase, the sender generates shares of the n secrets he holds and distributes them to the m servers. The sender does not intervene in the rest of the protocol. During the transfer phase, the receiver contacts K servers ($1 < K \leq m$) to collect enough shares to construct $\omega_{\sigma_1}, \omega_{\sigma_2}, \ldots, \omega_{\sigma_t}$.

As in other DOT protocols where only a subset of servers are contacted by the receiver (see [19, 20, 74]), it is also assumed that a mechanism prevents the receiver from contacting more than the specified number of servers in the transfer phase.

In the rest of this chapter, all computations are performed in the finite field $\mathbb{K}$. It is assumed that the inequality $p > \max(n, m)$ is satisfied. The set of indices of secrets ω_1, $\omega_2, \ldots, \omega_n$ held by $\mathcal{S}$ is denoted by $\mathcal{I}_\mathrm{n} = [\![1, n]\!]$. In addition, by an abuse of language, a polynomial $F = \sum_{i=0}^{i=\ell} a_i X^i$ of $\mathbb{K}_\ell[X]$ and its corresponding polynomial function $f : \mathbb{K} \longrightarrow \mathbb{K}, x \longmapsto \sum_{i=0}^{i=\ell} a_i x^i$ will not be differentiated (coefficients $a_0, a_1, \ldots, a_\ell$ belong to $\mathbb{K}$).

6.4 Description of the Protocol

The set-up phase is similar to the one described in Sect. 2.3.2 except that, to avoid a specific processing of ω_0, the form of the sparse polynomial generated by the sender is slightly modified:

$$Q(x, y_1, y_2, \ldots, y_n) = \alpha + \sum_{i=1}^{k-1} a_i x^i + \sum_{i=1}^{n} (\omega_i - \alpha) \times y_i \,,$$

where the coefficients α and a_i ($1 \leq i \leq k-1$) are elements randomly selected in $\mathbb{K}$.

In the transfer phase, the servers have the same behaviour as in the original protocol. On the other hand, the receiver:

- Selects $K = k + t - 1$ servers to contact instead of k servers.

- Generates t sets of n sharing polynomials instead of $n-1$ sharing polynomials. One set of polynomials is associated with a chosen secret. Since two distinct chosen secrets produce two secret inputs with two differences (two '1' in different positions as in the n-tuples $[\,1,0,0,\ldots,0\,]$ and $[\,0,1,0,0,\ldots,0\,]$), the receiver is able to use the same polynomials to share the $(n-2)$ '0' values in the same positions, which minimises the number of sharing polynomials to generate. From the K collected shares, the receiver prepares t sets of k shares ($k-1$ shares common to all sets) and interpolates t polynomials. From each of these t polynomials, the receiver is able to obtain one secret.

The resulting protocol is described in Fig. 6.1.

6.5 Security of the Protocol

6.5.1 Correctness

The receiver $\mathcal{R}$ sends an n-tuple $\mathbf{\Theta}_j = \left[\, Z_1^{(\sigma_1)}(j), Z_2^{(\sigma_1)}(j), \ldots, Z_n^{(\sigma_1)}(j) \,\right]$ to each server S_j $(1 \leq j \leq k-1)$. Thus, the response returned to $\mathcal{R}$ by S_j is

$$F_j(\mathbf{\Theta}_j) = \alpha + \sum_{i=1}^{k-1} a_i j^i + \sum_{i=1}^{n} (\omega_i - \alpha) Z_i^{(\sigma_1)}(j)\,.$$

By construction, for $j = 1, 2, \ldots, k-1$ and $\ell = 1, 2, \ldots, t$, $Z_i^{(\sigma_1)}(j) = Z_i^{(\sigma_\ell)}(j) = \lambda_{i,j}$. It follows that the interpolation of the k shares $F_j(\mathbf{\Theta}_j)$ $(j = 1, 2, \ldots, k-1, k-1+\ell)$ gives a polynomial of $\mathbb{K}_{k-1}[X]$,

$$G^{(\sigma_\ell)}(x) = \alpha + \sum_{i=1}^{k-1} a_i x^i + \sum_{i=1}^{n} (\omega_i - \alpha) Z_i^{(\sigma_\ell)}(x)\,.$$

The evaluation of $G^{(\sigma_\ell)}$ for $x = 0$ is therefore:

$$G^{(\sigma_\ell)}(0) = \alpha + \sum_{i=1}^{k-1} a_i 0^i + \sum_{i=1}^{n} (\omega_i - \alpha) Z_i^{(\sigma_\ell)}(0) = \omega_{\sigma_\ell}\,.$$

To conclude, the protocol is correct, i.e., security condition C_1' is satisfied.

6.5.2 Receiver's Privacy Against a Coalition of Servers

The indices $\sigma_\ell \in \mathcal{I}_t$ chosen by the receiver are represented under the form of n-tuples $\left[\, \delta_1^{\sigma_\ell}, \delta_2^{\sigma_\ell}, \ldots, \delta_n^{\sigma_\ell} \,\right]$ of private values, where $\delta_j^{\sigma_\ell} = 0$ or $\delta_j^{\sigma_\ell} = 1$. The receiver's input to

Let $\mathcal{I}_{\mathrm{m}}$ ($m > 1$) be a set of m distinct elements of $\mathbb{K}^*$ and S_j a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Input The sender $\mathcal{S}$ contributes with n secrets $\omega_1, \omega_2, \ldots, \omega_n \in \mathbb{K}$.
The receiver $\mathcal{R}$ chooses t indices $\sigma_1, \sigma_2, \ldots, \sigma_t$ of $\mathcal{I}_{\mathrm{n}}$, and contributes with t private n-tuples $\left[\, \delta_1^{\sigma}, \delta_2^{\sigma}, \ldots, \delta_n^{\sigma} \,\right]$, where $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_t$.

Output $\mathcal{R}$ receives $\omega_{\sigma_1}, \omega_{\sigma_2}, \ldots, \omega_{\sigma_t}$ while $\mathcal{S}$ receives nothing.

Set-up Phase

1 - *Preparation of a sharing polynomial.* The sender $\mathcal{S}$ generates a sparse $(n+1)$-variate polynomial Q defined by

$$Q\,(x, y_1, y_2, \ldots, y_n) = \alpha + \sum_{i=1}^{k-1} a_i x^i + \sum_{i=1}^{n} (\,\omega_i - \alpha\,) \times y_i\,,$$

where the coefficients α and a_i ($1 \leq i \leq k-1$) are randomly selected in $\mathbb{K}$.

2 - *Distribution of sharing polynomials.* Then, to each server S_j ($j \in \mathcal{I}_{\mathrm{m}}$), $\mathcal{S}$ transmits the n-variate polynomial F_j defined by

$$F_j\,(y_1, y_2, \ldots, y_n) = Q\,(j, y_1, y_2, \ldots, y_n)\ .$$

Transfer Phase

1 - *Selection of secret indices and servers.* The receiver $\mathcal{R}$ chooses the t secrets, ω_{σ_1}, $\omega_{\sigma_2}, \ldots, \omega_{\sigma_t}$, that she wishes to obtain. The set containing the indices of the t secrets is denoted by $\mathcal{I}_{\mathrm{t}} = \{\, \sigma_1, \sigma_2, \ldots, \sigma_t \,\} \subseteq \mathcal{I}_{\mathrm{n}}$. The receiver also selects a subset $\mathcal{I}_{\mathrm{K}} \subseteq \mathcal{I}_{\mathrm{m}}$, containing the $K = k + t - 1$ indices of the servers she intends to contact. Without loss of generality, it is assumed that these servers are $S_1, S_2, \ldots, S_K$.

2 - *Generation of the receiver's requests.* $\mathcal{R}$ builds n lists $\mathbf{\Lambda_i} = \left[\, \lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,k-1} \,\right]$ ($i \in \mathcal{I}_{\mathrm{n}}$) of $k-1$ elements randomly selected in $\mathbb{K}$. Then, for $\sigma_\ell \in \mathcal{I}_{\mathrm{t}}$, from each list $\mathbf{\Lambda_i}$, $\mathcal{R}$ interpolates a polynomial $Z_i^{(\sigma_\ell)} \in \mathbb{K}_{k-1}[X]$ from the k points of $\left\{ \left[\, 0, \delta_i^{\sigma_\ell} \,\right], \left[\, 1, \lambda_{i,1} \,\right], \left[\, 2, \lambda_{i,2} \,\right], \ldots, \left[\, k-1, \lambda_{i,k-1} \,\right] \right\}$.

3 - *Transmission of the requests to the servers.* $\mathcal{R}$ sends to each server S_j an n-tuple $\mathbf{\Theta}_j = \left[\, Z_1^{(\sigma_1)}(j), Z_2^{(\sigma_1)}(j), \ldots, Z_n^{(\sigma_1)}(j) \,\right]$ if $1 \leq j \leq k-1$ and an n-tuple $\mathbf{\Theta}_j = \left[\, Z_1^{(\sigma_{j-k+1})}(j), Z_2^{(\sigma_{j-k+1})}(j), \ldots, Z_n^{(\sigma_{j-k+1})}(j) \,\right]$ if $k \leq j \leq K$.

4 - *Transmission of the requests' responses to the receiver.* When a server S_j ($j \in \mathcal{I}_{\mathrm{K}}$) receives a request $\mathbf{\Theta}_j$, it replies with the share $F_j(\mathbf{\Theta}_j)$.

5 - *Construction of the requested secrets.* For each secret index $\sigma_\ell \in \mathcal{I}_{\mathrm{t}}$, $\mathcal{R}$ interpolates a univariate polynomial $G^{(\sigma_\ell)} \in \mathbb{K}_{k-1}[X]$ from the k values $F_j(\mathbf{\Theta}_j)$ ($1 \leq j \leq k-1$) and $F_{k-1+\ell}(\mathbf{\Theta}_{k-1+\ell})$. Then, $\mathcal{R}$ calculates the t secrets $\omega_{\sigma_\ell} = G^{(\sigma_\ell)}(0)$ ($\ell \in \mathcal{I}_{\mathrm{t}}$).

Figure 6.1: A One-Round t-out-of-n Distributed Oblivious Transfer Protocol

the protocol consists of shares of these values. That is, each server S_j ($j \in \mathcal{I}_{\mathrm{K}}$) receives for each of the n elements $\delta_i^{\sigma_\ell}$ a share produced by a (k, K)-threshold Shamir's secret sharing scheme [89] (see Sect. 2.3.1).

In order to breach the privacy of the receiver, a set of $k-1$ colluding servers should be able to determine at least one of the values $\delta_j^{\sigma_\ell}$. The set of $k-1$ collaborating servers, however, owns $k-1$ shares corresponding to each values $\delta_j^{\sigma_\ell}$ associated with a (k, K)-threshold Shamir's scheme. Due to the perfectness of Shamir's secret sharing scheme (see App. A), every set of $k-1$ shares provides the coalition with absolutely no information about the relevant secret.

It follows that security condition C_2' is satisfied with a security parameter $\lambda_{\mathrm{R}}' = k-1$.

6.5.3 Sender's Security

6.5.3.1 Sender's Security Against a Dishonest Receiver

During the transfer phase, $\mathcal{R}$ contacts K servers to collect shares. It is assumed that $\mathcal{R}$ does not follow the protocol and sends an n-tuple $\mathbf{\Theta}_j = \left[y_{j,1}, y_{j,2}, \ldots, y_{j,n} \right]$ of elements of $\mathbb{K}$ to the server S_j ($j \in \mathcal{I}_{\mathrm{K}}$). From the K collected shares, $\mathcal{R}$ is able to build a linear system of K equations in $k+n$ unknowns, namely $\alpha, a_1, a_2, \ldots, a_{k-1}, \omega_1, \omega_2, \ldots, \omega_n$.

This system may also be written under the matrix form $\boldsymbol{\mathcal{T}} \times \boldsymbol{U} = \boldsymbol{\Gamma}$, where

$$\boldsymbol{\mathcal{T}} = \begin{pmatrix} (1 - \sum_{i=1}^{n} y_{1,i}) & 1^1 & 1^2 & \ldots & 1^{k-1} & y_{1,1} & y_{1,2} & \ldots & y_{1,n} \\ (1 - \sum_{i=1}^{n} y_{2,i}) & 2^1 & 2^2 & \ldots & 2^{k-1} & y_{2,1} & y_{2,2} & \ldots & y_{2,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ (1 - \sum_{i=1}^{n} y_{K,i}) & K^1 & K^2 & \ldots & K^{k-1} & y_{K,1} & y_{K,2} & \ldots & y_{K,n} \end{pmatrix},$$

$$\boldsymbol{U} = \begin{pmatrix} \alpha \\ a_1 \\ a_2 \\ \ldots \\ a_{k-1} \\ \omega_1 \\ \omega_2 \\ \ldots \\ \omega_n \end{pmatrix} \text{ and } \boldsymbol{\Gamma} = \begin{pmatrix} G^{(1)}(0) \\ G^{(2)}(0) \\ \ldots \\ G^{(K)}(0) \end{pmatrix}.$$

To obtain r linear combinations of the secrets $\omega_1, \omega_2, \ldots, \omega_n$, $\mathcal{R}$ has to transform the system $\mathcal{T} \times \boldsymbol{U} = \boldsymbol{\Gamma}$ into an equivalent system $\mathcal{T}' \times \boldsymbol{U} = \boldsymbol{\Gamma}'$, where

$$\mathcal{T}' = \begin{pmatrix} A & B \\ 0 & D \end{pmatrix}$$

and $A \in \mathcal{M}_{K-r,k}(\mathbb{K})$, $B \in \mathcal{M}_{K-r,n}(\mathbb{K})$ and $D \in \mathcal{M}_{r,n}(\mathbb{K})$. In $\mathcal{T}$, one can observe a Vandermonde sub-matrix composed of the columns 2 to k:

$$\begin{pmatrix} 1^1 & 1^2 & \ldots & 1^{k-1} \\ 2^1 & 2^2 & \ldots & 2^{k-1} \\ \ldots & \ldots & \ldots & \ldots \\ k^1 & k^2 & \ldots & k^{k-1} \end{pmatrix}.$$

The rank of this sub-matrix is $k-1$, and so the number of rows of A is at least $k-1$. It follows that $K - r = k + t - 1 - r \geq k - 1$ and so $r \leq t$. Therefore, $\mathcal{R}$ cannot determine more than $r = t$ linear combinations of secrets.

To force the receiver to obtain information on no more than t secrets from the t linear combinations of secrets, the technique described by Naor and Pinkas [74] may be applied; in the set-up phase, the sender randomly selects n masks $c_1, c_2, \ldots, c_n \in \mathbb{K}^*$. Two polynomials Q_1 and Q_2 are generated (see Sect. 2.3.3), Q_1 to share the n masked secrets $c_i\omega_i$ and Q_2 to share the n masks c_i ($1 \leq i \leq n$). In the transfer phase, in response to the receiver's request $[\, Z_1(j), Z_2(j), \ldots, Z_n(j) \,]$, the contacted server S_j ($j \in \mathcal{I}_{\mathrm{K}}$) returns two shares: $Q_1(j, Z_1(j), Z_2(j), \ldots, Z_n(j))$ and $Q_2(j, Z_1(j), Z_2(j), \ldots, Z_n(j))$. From the collected $2K$ shares the receiver is able to determine exactly t masked secrets and their corresponding masks.

It follows that the protocol is secure against a dishonest receiver, i.e., security condition $C'_{4.1}$ is satisfied.

6.5.3.2 Sender's Security Against a Coalition of Servers Before the Protocol is Executed

Assuming the technique preventing the receiver from obtaining information on more than t secrets is used (see Sect. 6.5.3.1), each server S_j ($j \in \mathcal{I}_{\mathrm{m}}$) receives two n-variate polynomial functions from $\mathcal{S}$. More precisely, each server S_j receives a list of $2 \times (n+1)$ elements: $\alpha_1 + \sum_{i=1}^{k-1} a_{1,j} j^i$, $\alpha_2 + \sum_{i=1}^{k-1} a_{2,j} j^i$, $c_1\omega_1 - \alpha_1$, $c_2\omega_2 - \alpha_1, \ldots, c_n\omega_n - \alpha_1$, and $c_1 - \alpha_2, c_2 - \alpha_2, \ldots, c_n\alpha_2$. The first two elements can be considered as shares generated by (k, m)-threshold Shamir's secret sharing schemes whereas the other elements can be viewed as $2n$ secrets masked by α_1 and α_2, the masks α_1 and α_2 being unknown. To

determine α_1 or α_2, a coalition of servers would need to obtain the corresponding sharing polynomial which is of degree at most $k-1$. So, the coalition should contain at least k members. In addition, although each server is able to determine linear combinations of masks $c_1, c_2, \ldots, c_n$ they cannot obtain linear combinations of the secrets $\omega_1, \omega_2, \ldots, \omega_n$, as shown by Cheong, Koshiba, and Nishiyama [29]. Note that in this scenario, the use of the technique described in Sect. 6.5.3.1 makes the additional masking of secrets described in Blundo et al.'s DOT sub-protocol [19, 20] unnecessary (see Sect. 3.5.2).

Furthermore, in this scenario there is no advantage for the receiver to collude with $k-1$ servers to breach the sender's security, since the receiver has no input to contribute in an attack.

To conclude, a coalition of the receiver with $k-1$ servers cannot obtain any information on the secrets held by the sender before the transfer phase is executed, i.e., security condition C'_3 is satisfied with the security parameter $\lambda'_S = k-1$.

6.5.3.3 Sender's Security Against a Coalition of Servers After the Protocol is Executed

As in the original protocol, the security of the sender is not guaranteed if servers are corrupted by the receiver once the protocol has been executed. Actually, if the receiver corrupts only one server, she is able to obtain all secrets. Indeed, any corrupt server S_j ($j \in \mathcal{I}_m$) is able to provide the receiver with the elements $c_1\omega_1-\alpha_1, c_2\omega_2-\alpha_1, \ldots, c_n\omega_n-\alpha_1$, $c_1-\alpha_2, c_2-\alpha_2, \ldots, c_n-\alpha_2$. Moreover, the execution of the protocol gives the receiver t masks $c_{\sigma_1}, c_{\sigma_2}, \ldots, c_{\sigma_t}$, as well as t secrets, $\omega_{\sigma_1}, \omega_{\sigma_2}, \ldots, \omega_{\sigma_t}$. Consequently, simple subtractions allow the receiver to determine α_1 and α_2 and hence all masks and masked secrets. Then, simple divisions allow the receiver to calculate all the secrets $\omega_1, \omega_2, \ldots, \omega_n$.

It follows that security condition $C'_{4.2}$ cannot be guaranteed for a security parameter $\lambda'_C > 0$.

6.6 Efficiency Consideration

In this section, the efficiency of the protocol presented in this chapter is compared to the efficiency of Blundo et al.'s protocol [19, 20] regarding the number of exchanges amongst the different parties. To obtain t secrets, the receiver has to run the proposed protocol once and Blundo et al.'s protocol t times. The results are shown in Table 6.1.

To improve the fairness of the comparison, the masks of the sub-protocol described in [19, 20] are not taken into account, i.e., the $2 \times t \times m \times (n-1)$ shares distributed by the sender to the servers and the $2 \times t \times k$ shares collected by the receiver (see Sect. 3.5.2).

Table 6.1: Efficiency of t-out-of-n Distributed Oblivious Transfer Protocols

Parties	**Blundo et al.'s Protocol**	**Proposed Protocol**
Sender $\mathcal{S} \rightarrow$ Servers S_j	$t \times 2 \times m \times n$ elements of $\mathbb{K}$, n polynomial coefficients per server per run per secret to be obtained by $\mathcal{R}$	$2 \times m \times (n+1)$ elements of $\mathbb{K}$, $n+1$ polynomial coefficients per server per set of shares returned to $\mathcal{R}$
Receiver $\mathcal{R} \rightarrow$ Servers S_j	$t \times k \times (n-1)$ elements of $\mathbb{K}$, 1 share per private value per server per secret to be obtained by $\mathcal{R}$	$(k+t-1) \times n$ elements of $\mathbb{K}$, 1 share per private value per server
Servers $S_j \rightarrow$ Receiver $\mathcal{R}$	$t \times 2 \times k$ elements of $\mathbb{K}$, 1 share per server per set of shares returned to $\mathcal{R}$ per secret to be obtained by $\mathcal{R}$	$2 \times (k+t-1)$ elements of $\mathbb{K}$, 1 share per server per set of shares returned to $\mathcal{R}$

It is observed that in the set-up phase, the sender $\mathcal{S}$ transmits around t fewer shares to the servers S_i ($i \in \mathcal{I}_m$) in the proposed protocol than in Blundo et al.'s one.

Furthermore, the transfer phase of the proposed protocol is around $e = \frac{kt}{k+t-1}$ times more efficient than the transfer phase of the original protocol executed t times. Note that e belongs to the interval $[\,1, k\,] \subset \mathbb{Q}$ with the case $e = 1$ corresponding to the receiver's request for $t = 1$ secret only.

6.7 Conclusion

In this chapter was introduced the first unconditionally secure distributed oblivious transfer protocol allowing a receiver to obtain t out of n secrets in one round only. The proposed protocol is significantly more efficient than t runs of DOT protocols allowing the receiver to obtain one secret per run.

Note that the list of the t secrets the receiver wishes to obtain needs to be established before the transfer phase starts. The two DOT protocols presented in the next chapter also allow a receiver to obtain t out of n secrets. Additionally, these two protocols enable the receiver to select a secret based upon the value of the secrets previously obtained.

7

Adaptive Distributed Oblivious Transfer Protocols

A *distributed oblivious transfer* protocol allows a receiver to choose one out of the n secrets held by a sender, by contacting k servers. Once the protocol has been executed, the sender learns nothing about the receiver's choice and the receiver does not have information on the secrets she did not choose.

The current unconditionally secure polynomial interpolation-based distributed oblivious transfer protocols all suffer the same weakness: security cannot be guaranteed if the receiver, after she has obtained a secret, wishes to obtain a second secret by contacting, again, k servers.

In this chapter, two unconditionally secure distributed oblivious transfer protocols are proposed. They allow a receiver to sequentially obtain t secrets, more efficiently than by simply executing a distributed oblivious transfer protocol t times. In the first protocol, the number of secrets the receiver can obtain is limited, in contrast to the second protocol. However, in the second protocol, the contacted servers need to communicate with each other. In addition, both protocols guarantee the sender's and the receiver's privacy against a coalition of $k - 1$ servers.

7.1 Introduction

A *distributed oblivious transfer* (DOT) protocol enables a *receiver* to contact 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. Polynomial interpolation-based DOT

protocols (e.g., [19, 20, 29, 74, 78]) are composed of two phases: a *set-up phase*, allowing the sender to generates shares of the n secrets he holds and to distributes them to m servers, and a *transfer phase*, enabling the receiver to contact k servers ($1 < k \leq m$) to collect enough shares to determine the chosen secret. The sender only intervenes in the set-up phase. The receiver obtains the chosen secret in the single round of the transfer phase of the protocol, a round being defined as a set of consistent requests/responses exchanged between the receiver and k servers. If the receiver wishes to obtain t secrets, she may use the t-out-of-n DOT protocol presented in the previous chapter. However, if the choice of one secret depends on the secrets already obtained (the protocol is said to be adaptive [77]), the t-out-of-n DOT protocol presented in Chap. 6 is unsuitable and other DOT protocols have to be executed t times, resulting in a low overall efficiency.

Recently, Jiang, Li, and Li [65] have proposed an unconditionally secure DOT protocol with adaptive queries. However, their protocol consists in the repetition of the transfer phase of Nikov, Nikova, Preneel, and Vanderwalle's protocol [78]. This technique does not guarantee the protocol's security against a dishonest — but not malicious — receiver (see App. D).

In this chapter, two unconditionally secure DOT protocols with adaptive queries are introduced; both protocols are adapted from Blundo, D'Arco, De Santis, and Stinson's polynomial interpolation-based DOT protocol [19, 20].

In the first one, the servers receive a list of t tokens from the sender and consume a token of the list for each query prepared by the receiver. Thus, the receiver cannot obtain more than t secrets. This model is well adapted to a single receiver, but for several receivers, a second protocol is proposed; this protocol accepts an unlimited number of queries. In this second protocol, servers are allowed to communicate with each other and for each query, generate new *ad hoc* secret sharing polynomials.

The chapter is organized as follows. In Sect. 7.2 a few notations and definitions are introduced. In Sect. 7.3, security and communication models are presented. The first DOT protocol is described in Sect. 7.4 and the second one in Sect. 7.5. The communication and computation performance of both protocols is discussed in Sect. 7.6. Finally, a brief conclusion is given in Sect. 7.7.

7.2 Preliminaries

The settings of the two DOT protocols described in this chapter encompass a sender $\mathcal{S}$ who owns n secrets $\omega_1, \omega_2, \ldots, \omega_n$ ($n > 1$) selected in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power), a receiver $\mathcal{R}$ who wishes to learn t secrets ($1 \leq t < n$), and m servers S_i ($m > 1$ and $i \in \mathcal{I}_\mathrm{m} \subseteq \mathbb{K}^*$, where $|\mathcal{I}_\mathrm{m}| = m$). All operations are executed in $\mathbb{K}$ and it is

assumed that $p > \max(n, m)$. Also, the set of indices of secrets $\omega_1, \omega_2, \ldots, \omega_n$ held by $\mathcal{S}$ is denoted by $\mathcal{I}_{\mathrm{n}} = [\![1, n]\!]$.

7.3 Model

Each protocol is composed of a set-up phase and t transfer phases (run 1 to run t). The receiver wishes to learn a secret ω_{σ_ℓ} ($\sigma_\ell \in \mathcal{I}_{\mathrm{n}}$) in run ℓ.

During the set-up phase, the sender generates shares of the n secrets he holds and distributes them to the m servers. The sender does not intervene in the rest of the protocol. In run ℓ, the receiver has to contact k servers ($1 < k \leq m$) to collect enough shares to construct ω_{σ_ℓ}.

7.3.1 Security Model

The objective is to propose unconditionally secure adaptive DOT protocols with the same level of security as in Blundo et al.'s protocol [19, 20]. Of course, because of the adaptive feature of the proposed protocols, it is necessary to adjust some of the security properties, to extend the original security to t secrets; actually, the same security conditions and parameters as those described in Sect. 6.2 are adopted. Thus, it is shown that similarly to other oblivious transfer protocols, the proposed protocols guarantee the following properties:

- When all participants follow the protocol, $\mathcal{R}$ obtains the selected secrets (correctness is guaranteed, i.e., security condition C_1' is satisfied).
- Assuming the servers and $\mathcal{S}$ are honest, even if $\mathcal{R}$ does not follow the protocol, she cannot obtain information on the secrets she did not choose (sender's security is guaranteed, i.e., security condition $C_{4.1}'$ is satisfied).
- Assuming the servers and $\mathcal{R}$ are honest, $\mathcal{S}$ who may not follow the protocol, cannot obtain any information on the receiver's selection. Because $\mathcal{S}$ is only involved in the set-up phase and other participants are not supposed to communicate with him in the runs, this property is *de facto* guaranteed.

In addition, because of the distributed setting, the protocols are required to satisfy the following properties:

- The receiver's privacy against a passive coalition of $k - 1$ servers (security condition C_2', with $\lambda_{\mathrm{R}}' = k - 1$ is satisfied).

- The sender's security against a passive coalition of the receiver and $k-1$ servers, before the first run is started (security condition C_3', with $\lambda_S' = k-1$ is satisfied).

As in the original protocol, the proposed protocols do not guarantee the sender's security against a 'greedy' receiver who, once a protocol is completed and she has obtained t secrets, would corrupt one server. With the additional information collected from the corrupted server, the receiver is able to determine all secrets (security condition C_4' with $\lambda_C > 0$ is not satisfied).

7.3.2 Communication Model

Both protocols require the availability of private unicast communication channels between the sender and the servers and between the receiver and the servers. It is assumed 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 (see Sect. 2.2.1).

In addition, in the first protocol it is assumed that a broadcast channel is available and in the second protocol, it is assumed that a secure unicast communication channel between any two servers is available.

As in other DOT protocols where only a subset of servers is contacted by the receiver [19, 20, 74], it is assumed that a mechanism prevents the receiver from contacting more than k servers in each run.

7.4 First Distributed Oblivious Transfer Protocol with Adaptive Queries

In this section, the first protocol is described and its security proved.

7.4.1 Description

The first protocol is a basic adaptation of Blundo et al.'s one-round polynomial-based DOT protocol [19, 20] (see Fig. 2.1) where the transfer phase is executed t times.

The major characteristics of the adapted protocol are:

- In the set-up phase, the sender generates a list of polynomials $Q^{(\ell)}$ $(1 \leq \ell \leq t)$ instead of a single polynomial Q.
- In run ℓ, the contacted servers use the polynomial $Q^{(\ell)}$ to respond to the receiver's query.

To avoid a specific processing of the first secret as in the original protocol, the sparse polynomial generated by the sender is slightly transformed; in the set-up phase, the sender selects a random element $\alpha \in \mathbb{K}$ and generates t quasi-random polynomials $B_0^{(\ell)}$ $(1 \leq \ell \leq t)$, of $\mathbb{K}_{k-1}[X]$, such that $B_0^{(\ell)}(0) = \alpha$. The sender is then able to build the t $(n+1)$-variate polynomials

$$Q^{(\ell)}(x, y_1, y_2, \ldots, y_n) = B_0^{(\ell)}(x) + \sum_{i=1}^{n} (\omega_i - \alpha) \times y_i \,.$$

To transmit a list of t polynomials

$$Q^{(1)}(j, y_1, y_2, \ldots, y_n), Q^{(2)}(j, y_1, y_2, \ldots, y_n), \ldots, Q^{(t)}(j, y_1, y_2, \ldots, y_n)$$

to a server S_j, the sender just has to send to S_j the values $B_0^{(1)}(j), B_0^{(2)}(j), \ldots, B_0^{(t)}(j)$, $\omega_1 - \alpha, \omega_2 - \alpha, \ldots, \omega_n - \alpha$.

A server contacted by the receiver needs to know what polynomial $Q^{(\ell)}$ to use, i.e., what is the current run number ℓ. Therefore, a mechanism is added, to synchronize the contacted servers thanks to a broadcast channel. Before sending requests, the receiver broadcasts the number of the next run as well as the list of servers she intends to contact. On reception of the next run number ℓ, a server with a local run number ℓ' checks if $\ell' < \ell \leq t$. If the double inequality is satisfied, then the server updates its local run number to ℓ. Otherwise it shuts down since the receiver has either reused a run number or tried to execute more than t runs. To prevent the receiver from taking advantage of a disconnected network and to use the same run number for two queries, it is necessary that $k > m/2$. So, if a server finds out that the broadcast list contains $m/2$ or fewer servers, it shuts down. Moreover, a server which receives a request from the receiver has to check that it belongs to the broadcast list. Otherwise, it shuts down too.

The DOT protocol introduced by Blundo et al. [19], including the above modifications, is described in Fig. 7.1.

7.4.2 Security

It is shown that the protocol is correct and guarantees the sender's security. In addition, the two scenarios involving a coalition of active parties are considered.

7.4.2.1 Correctness.

In run ℓ, a contacted server S_j $(j \in \mathcal{I}^{(\ell)})$ uses the $n+1$ elements of $\mathbb{K}$, $B_0^{(\ell)}(j)$, $\omega_1 - \alpha$, $\omega_2 - \alpha, \ldots, \omega_n - \alpha$, to prepare the response $s_j^{(\ell)}$. We have:

$$s_j^{(\ell)} = B_0^{(\ell)}(j) + \sum_{i=1}^{n} (\omega_i - \alpha) \times D_i^{(\ell)}(j) \,.$$

Let $\mathcal{I}_{\mathrm{m}}$ ($m > 1$) be a set of m distinct elements of $\mathbb{K}^*$ and S_j a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Set-up Phase

1. The sender $\mathcal{S}$ selects a random element $\alpha \in \mathbb{K}$ and generates t quasi-random polynomials $B_0^{(\ell)} = \sum_{i=1}^{k-1} a_i^{(\ell)} X^i + \alpha$ ($1 \leq \ell \leq t$).

2. Then, to each server S_j ($j \in \mathcal{I}_{\mathrm{m}}$), $\mathcal{S}$ transmits the $t + n$ values $B_0^{(1)}(j)$, $B_0^{(2)}(j), \ldots, B_0^{(t)}(j), \omega_1 - \alpha, \omega_2 - \alpha, \ldots, \omega_n - \alpha$, as well as a local current run number initialized to 0 (it is necessary that $t \leq p - 1$).

Run ℓ **(**$1 \leq \ell \leq t$**)**

1. The receiver $\mathcal{R}$ chooses the index $e^{(\ell)}$ of one secret and generates n quasi-random polynomials $Z_i^{(\ell)}$ ($i \in \mathcal{I}_{\mathrm{n}}$), of degree at most $k - 1$, such that $\left[Z_1^{(\ell)}(0), Z_2^{(\ell)}(0), \ldots, Z_n^{(\ell)}(0) \right] = \left[\delta_1^{e^{(\ell)}}, \delta_2^{e^{(\ell)}}, \ldots, \delta_n^{e^{(\ell)}} \right]$.

2. Then, $\mathcal{R}$ selects a subset $\mathcal{I}^{(\ell)} \subseteq \mathcal{I}_{\mathrm{m}}$ of k indices and broadcasts the current run number ℓ as well as the list $\mathcal{I}^{(\ell)}$.

3. On reception of ℓ and $\mathcal{I}^{(\ell)}$, a server S_j ($j \in \mathcal{I}_{\mathrm{m}}$) with a local run number ℓ' checks that $\ell \leq t$ and $\ell' < \ell$. If the two inequalities are satisfied, the local run number is updated with ℓ, otherwise, S_j shuts down.

4. Then, $\mathcal{R}$ sends to each server S_j ($j \in \mathcal{I}^{(\ell)}$) the request

$$\left[Z_1^{(\ell)}(j), Z_2^{(\ell)}(j), \ldots, Z_n^{(\ell)}(j) \right] .$$

5. When a server S_j receives such a request, it first checks that it is in the list $\mathcal{I}^{(\ell)}$. If this condition is satisfied, S_j replies with the share $s_j^{(\ell)} = B_0^{(\ell)}(j) + \sum_{i=1}^{n} (\omega_i - \alpha) \times Z_i^{(\ell)}(j)$. Otherwise, S_j shuts down.

6. After receiving k responses $s_j^{(\ell)}$ ($j \in \mathcal{I}^{(\ell)}$), $\mathcal{R}$ interpolates a univariate polynomial $R^{(\ell)}$ from the k points $\left[j, s_j^{(\ell)} \right]$ and calculates the chosen secret: $\omega_{e^{(\ell)}} = R^{(\ell)}(0)$.

Figure 7.1: Distributed Oblivious Transfer Protocol with Adaptive Queries (Limited Number of Queries)

The polynomial $B_0^{(\ell)}$ is of degree at most $k - 1$, as well as the polynomials $D_i^{(\ell)}$ ($1 \leq i \leq n$). It follows that the degree of the polynomial $R = B_0^{(\ell)} + \sum_{i=1}^{n} (\omega_i - \alpha) \times D_i^{(\ell)}$ is at most $k - 1$. With the k values $s_j^{(\ell)}$, the receiver is therefore able to interpolate this polynomial R. Thus, the receiver can calculate

$$
\begin{aligned}
R^{(\ell)}(0) &= \left(B_0^{(\ell)} + \sum_{i=1}^{n} (\omega_i - \alpha) \times D_i^{(\ell)} \right)(0) \\
&= B_0^{(\ell)}(0) + \sum_{i=1}^{n} (\omega_i - \alpha) \times D_i^{(\ell)}(0) \\
&= \alpha + \sum_{i=1}^{n} (\omega_i - \alpha) \times \delta_i^{e^{(\ell)}}
\end{aligned}
$$

It follows that $R^{(\ell)}(0) = \alpha + (\omega_{e^{(\ell)}} - \alpha) = \omega_{e^{(\ell)}}$. Therefore, the protocol is correct (security condition C_1' is satisfied).

7.4.2.2 Sender's Security.

It is assumed that $\mathcal{R}$ does not follow the protocol and in run ℓ ($1 \leq \ell \leq t$) sends an n-tuple $\boldsymbol{y_j^{(\ell)}} = \left[y_{i_j^{(\ell)},1}, y_{i_j^{(\ell)},2}, \ldots, y_{i_j^{(\ell)},n} \right]$ of elements of $\mathbb{K}$ to the server $S_{i_j^{(\ell)}}$ ($i_j^{(\ell)} \in \mathcal{I}^{(\ell)} = \{ i_1^{(\ell)}, i_2^{(\ell)}, \ldots, i_k^{(\ell)} \}$). After t runs have been executed, $\mathcal{R}$ has collected kt shares and is able to build a linear system of kt equations in $(k-1)t + n + 1$ unknowns, namely $a_1^{(1)}$, $a_2^{(1)}, \ldots, a_{k-1}^{(1)}, a_1^{(2)}, a_2^{(2)}, \ldots, a_{k-1}^{(2)}, \ldots, a_1^{(t)}, a_2^{(t)}, \ldots, a_{k-1}^{(t)}, \alpha, \omega_1, \omega_2, \ldots, \omega_n$.

If we denote

$$
\mathcal{V}_{r,c}^{(\ell)} = \begin{pmatrix} (i_1^{(\ell)})^1 & (i_1^{(\ell)})^2 & \ldots & (i_1^{(\ell)})^c \\ (i_2^{(\ell)})^1 & (i_2^{(\ell)})^2 & \ldots & (i_2^{(\ell)})^c \\ \vdots & \vdots & & \vdots \\ (i_r^{(\ell)})^1 & (i_r^{(\ell)})^2 & \ldots & (i_r^{(\ell)})^c \end{pmatrix}, \quad \boldsymbol{z_r^{(\ell)}} = \begin{pmatrix} z_1^{(\ell)} = \left(1 - \sum_{j=1}^{n} (\omega_{i_1^{(\ell)}} - \alpha) D_j^{(\ell)}(i_1^{(\ell)}) \right) \\ z_2^{(\ell)} = \left(1 - \sum_{j=1}^{n} (\omega_{i_2^{(\ell)}} - \alpha) D_j^{(\ell)}(i_2^{(\ell)}) \right) \\ \vdots \\ z_r^{(\ell)} = \left(1 - \sum_{j=1}^{n} (\omega_{i_r^{(\ell)}} - \alpha) D_j^{(\ell)}(i_r^{(\ell)}) \right) \end{pmatrix},
$$

$$
\boldsymbol{\mathcal{Y}_r^{(\ell)}} = \begin{pmatrix} \boldsymbol{y_1^{(\ell)}} \\ \boldsymbol{y_2^{(\ell)}} \\ \vdots \\ \boldsymbol{y_r^{(\ell)}} \end{pmatrix}, \quad \boldsymbol{s^{(\ell)}} = \begin{pmatrix} s_1^{(\ell)} \\ s_2^{(\ell)} \\ \vdots \\ s_k^{(\ell)} \end{pmatrix} \quad \text{and} \quad \boldsymbol{a^{(\ell)}} = \begin{pmatrix} a_1^{(\ell)} \\ a_2^{(\ell)} \\ \vdots \\ a_{k-1}^{(\ell)} \end{pmatrix},
$$

the system built by $\mathcal{R}$ may be written under the matrix form $\boldsymbol{\mathcal{T}} \times \boldsymbol{U} = \boldsymbol{S}$, where:

$$\boldsymbol{\mathcal{T}} = \begin{pmatrix} \mathcal{V}_{k,k-1}^{(1)} & 0 & \dots & 0 & \boldsymbol{z_k^{(1)}} & \boldsymbol{\mathcal{Y}_k^{(1)}} \\ 0 & \mathcal{V}_{k,k-1}^{(2)} & \dots & 0 & \boldsymbol{z_k^{(2)}} & \boldsymbol{\mathcal{Y}_k^{(2)}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \mathcal{V}_{k,k-1}^{(t)} & \boldsymbol{z_k^{(t)}} & \boldsymbol{\mathcal{Y}_k^{(t)}} \end{pmatrix}, \quad \boldsymbol{U} = \begin{pmatrix} \boldsymbol{a^{(1)}} \\ \boldsymbol{a^{(2)}} \\ \vdots \\ \boldsymbol{a^{(t)}} \\ \alpha \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{pmatrix} \quad \text{and} \quad \boldsymbol{S} = \begin{pmatrix} \boldsymbol{s^{(1)}} \\ \boldsymbol{s^{(2)}} \\ \vdots \\ \boldsymbol{s^{(t)}} \end{pmatrix}.$$

To obtain r linear combinations of the secrets $\alpha, \omega_1, \omega_2, \dots, \omega_n, \mathcal{R}$ has to transform the system $\boldsymbol{\mathcal{T}} \times \boldsymbol{U} = \boldsymbol{S}$ into an equivalent system $\boldsymbol{\mathcal{T}'} \times \boldsymbol{U} = \boldsymbol{S'}$, where

$$\boldsymbol{\mathcal{T}'} = \begin{pmatrix} \mathcal{A}_1 & \mathcal{A}_2 \\ 0 & \mathcal{A}_3 \end{pmatrix}$$

and $\mathcal{A}_1 \in \mathcal{M}_{kt-r,(k-1)t}(\mathbb{K})$, $\mathcal{A}_2 \in \mathcal{M}_{kt-r,n+1}(\mathbb{K})$ and $\mathcal{A}_3 \in \mathcal{M}_{r,n+1}(\mathbb{K})$.

By moving each k-th row of $\mathcal{T}$ at the bottom of the matrix, we obtain a new matrix equivalent to $\mathcal{T}$:

$$\begin{pmatrix} \mathcal{V}_{k-1,k-1}^{(1)} & 0 & \dots & 0 & \boldsymbol{z_{k-1}^{(1)}} & \boldsymbol{\mathcal{Y}_{k-1}^{(1)}} \\ 0 & \mathcal{V}_{k-1,k-1}^{(2)} & \dots & 0 & \boldsymbol{z_{k-1}^{(2)}} & \boldsymbol{\mathcal{Y}_{k-1}^{(2)}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \mathcal{V}_{k-1,k-1}^{(t)} & \boldsymbol{z_{k-1}^{(t)}} & \boldsymbol{\mathcal{Y}_{k-1}^{(t)}} \\ (i_k^{(1)})^1 & (i_k^{(1)})^2 & \dots & (i_k^{(1)})^{k-1} & z_k^{(1)} & \boldsymbol{y_k^{(1)}} \\ (i_k^{(2)})^1 & (i_k^{(2)})^2 & \dots & (i_k^{(2)})^{k-1} & z_k^{(2)} & \boldsymbol{y_k^{(2)}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (i_k^{(t)})^1 & (i_k^{(t)})^2 & \dots & (i_k^{(t)})^{k-1} & z_k^{(t)} & \boldsymbol{y_k^{(t)}} \end{pmatrix}.$$

In this matrix, we observe t Vandermonde submatrices $\mathcal{V}_{k-1,k-1}^{(\ell)}$ $(1 \leq \ell \leq t)$ equivalent to identity matrices $\mathcal{I}_{k-1,k-1}$. It follows that $\mathcal{T}$ is equivalent to the matrix:

$$\mathcal{T}' = \begin{pmatrix} 1 & 0 & \dots & 0 & * & * \\ 0 & 1 & \dots & 0 & * & * \\ \vdots & \vdots & \ddots & \vdots & \vdots & * \\ 0 & 0 & \dots & 1 & * & * \\ (i_k^{(1)})^1 & (i_k^{(1)})^2 & \dots & (i_k^{(1)})^{k-1} & z_k^{(1)} & \boldsymbol{y_k^{(1)}} \\ (i_k^{(2)})^1 & (i_k^{(2)})^2 & \dots & (i_k^{(2)})^{k-1} & z_k^{(2)} & \boldsymbol{y_k^{(2)}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (i_k^{(t)})^1 & (i_k^{(t)})^2 & \dots & (i_k^{(t)})^{k-1} & z_k^{(t)} & \boldsymbol{y_k^{(t)}} \end{pmatrix}.$$

The rank of the upper left corner $(k-1)t \times (k-1)t$ sub-matrix is $(k-1)t$, and so the number of rows of $\mathcal{A}_1$ is at least $(k-1)t$. It follows that $kt - r \geq (k-1)t$ and so $r \leq t$. Therefore, $\mathcal{R}$ cannot determine more than $r = t$ linear combinations of secrets.

To force $\mathcal{R}$ to obtain information on no more than t secrets from the t linear combinations of secrets, the technique described by Naor and Pinkas [74] may be applied; in the set-up phase, $\mathcal{S}$ prepares a set of n random masks $c_1, c_2, \dots, c_n \in \mathbb{K}^*$. For each run ℓ, two polynomials $Q_1^{(\ell)}$ and $Q_2^{(\ell)}$ are generated (see Sect. 2.3.3), $Q_1^{(\ell)}$ to share the n masked secrets $c_i\omega_i$ and $Q_2^{(\ell)}$ to share the n masks c_i ($i \in \mathcal{I}_n$). In run ℓ, in response to the receiver's request $\left(D_1^{(\ell)}(j), D_2^{(\ell)}(j), \dots, D_n^{(\ell)}(j)\right)$, the contacted server S_j ($j \in \mathcal{I}^{(\ell)}$) returns two shares: $Q_1^{(\ell)}(j, D_1^{(\ell)}(j), D_2^{(\ell)}(j), \dots, D_n^{(\ell)}(j))$ and $Q_2^{(\ell)}(j, D_1^{(\ell)}(j), D_2^{(\ell)}(j), \dots, D_n^{(\ell)}(j))$. In each run, from the collected $2k$ shares, $\mathcal{R}$ is able to determine exactly one masked secret and its corresponding mask, and finally the chosen secret. Note that the use of this technique makes the additional masking of secrets described in Blundo et al.'s DOT sub-protocol [19, 20] unnecessary.

Thus, the sender's security is guaranteed (security condition $C'_{4.1}$ is satisfied).

7.4.2.3 Receiver's Privacy against a Coalition of Servers.

In run ℓ, the index $e^{(\ell)}$ chosen by the receiver is represented under the form of an n-tuple $\left[\delta_1^{e^{(\ell)}}, \delta_2^{e^{(\ell)}}, \dots, \delta_n^{e^{(\ell)}}\right]$ of private values. The receiver's input to the protocol consists of shares — generated by a k-threshold Shamir's scheme — of these values. This is achieved by generating n quasi-random polynomials $D_1^{(\ell)}, D_2^{(\ell)}, \dots, D_n^{(\ell)}$, of degree at most $k-1$, such that $D_i^{(\ell)}(0) = \delta_i^{e^{(\ell)}}$.

In order to breach the privacy of the receiver, a set of $k-1$ colluding servers should be able to determine at least one of the values $\delta_i^{e^{(\ell)}}$. The set of $k-1$ collaborating servers, however, owns $k-1$ shares corresponding to each values $\delta_i^{e^{(\ell)}}$ associated with a k-threshold Shamir's secret sharing scheme. Due to the perfectness of Shamir's secret sharing scheme (see App. A), every set of $k-1$ shares provides the coalition with absolutely no information

about the relevant secret. That is, the inputs of the receiver guarantee her privacy (security condition C_2' with $\lambda_{\mathrm{R}}' = k - 1$ is satisfied).

7.4.2.4 Sender's Security against a Coalition of the Receiver and Servers before the First Run.

Assuming the technique preventing the receiver from obtaining information on more than t secrets is used (see Sender's Security above), each server S_j ($j \in \mathcal{I}_{\mathrm{m}}$) receives $2t$ n-variate polynomial functions from $\mathcal{S}$. More precisely, each server S_j receives a list of $2 \times (n + t)$ elements: $B_{1,0}^{(\ell)} = \alpha_1 + \sum_{i=1}^{k-1} a_{1,j}^{(\ell)} j^i$, $B_{2,0}^{(\ell)} = \alpha_2 + \sum_{i=1}^{k-1} a_{2,j}^{(\ell)} j^i$, $c_1\omega_1 - \alpha_1$, $c_2\omega_2 - \alpha_1, \ldots, c_n\omega_n - \alpha_1$ and $c_1 - \alpha_2, c_2 - \alpha_2, \ldots, c_n - \alpha_2$. The first $2t$ elements can be considered as shares generated by (k, m)-threshold Shamir's secret sharing schemes whereas the other elements can be viewed as n secrets masked by α_1 and α_2, the masks α_1 and α_2 being unknown. To determine α_1 or α_2, a coalition of servers would need to obtain the corresponding sharing polynomial which is of degree at most $k - 1$. So, the coalition should contain at least k members. In addition, although each server is able to determine linear combinations of masks $c_1, c_2, \ldots, c_n$ they cannot obtain linear combinations of the secrets $\omega_1, \omega_2, \ldots, \omega_n$, as shown by Cheong, Koshiba, and Nishiyama [29].

Note that in this scenario, there is no advantage for the receiver to collude with $k - 1$ servers to breach the sender's security, since the receiver has no input to contribute in an attack.

To conclude, a coalition of the receiver with $k-1$ servers cannot obtain any information on the secrets held by the sender before the first run is executed (security condition C_3' with $\lambda_{\mathrm{S}}' = k - 1$ is satisfied).

7.5 Second Distributed Oblivious Transfer Protocol with Adaptive Queries

In this section, the second protocol is described and its security proved.

7.5.1 Description

The major problem with the current unconditionally secure polynomial-based DOT protocols [19, 20, 74, 78] is that when the receiver sends the same request to a server in two different runs, she obtains the same response. Thus, she is able to reuse responses from previous queries, which allows her to request additional shares and obtain additional secrets. The key idea of the second protocol is to force a server contacted in different runs,

Let $\mathcal{I}_{\mathrm{m}}$ ($m > 1$) be a set of m distinct elements of $\mathbb{K}^*$ and S_j a server such that $j \in \mathcal{I}_{\mathrm{m}}$.

Set-up Phase

1. The sender $\mathcal{S}$ selects a random element $\alpha \in \mathbb{K}$. Then, $\mathcal{S}$ generates a quasi-random polynomial B_0, of degree at most $k-1$, such that $B_0(0) = \alpha$.
2. Then, to each server S_j ($j \in \mathcal{I}_{\mathrm{m}}$), $\mathcal{S}$ transmits $B_0(j)$ and the n elements of $\mathbb{K}$, $\omega_1 - \alpha, \omega_2 - \alpha, \ldots, \omega_n - \alpha$.

Run ℓ ($\ell \geq 1$)

1. The receiver $\mathcal{R}$ chooses the index $e^{(\ell)}$ of the secret she wishes to obtain. She also generates n quasi-random polynomials $Z_i^{(\ell)}$ ($i \in \mathcal{I}_{\mathrm{n}}$) of $\mathbb{K}_{k-1}[X]$, such that $\left[Z_1^{(\ell)}(0), Z_2^{(\ell)}(0), \ldots, Z_n^{(\ell)}(0) \right] = \left[\delta_1^{e^{(\ell)}}, \delta_2^{e^{(\ell)}}, \ldots, \delta_n^{e^{(\ell)}} \right]$.
2. Then, $\mathcal{R}$ selects a subset $\mathcal{I}^{(\ell)} \subseteq \mathcal{I}_{\mathrm{m}}$ of k indices and sends to each server S_j ($j \in \mathcal{I}^{(\ell)}$) a request $\left[Z_1^{(\ell)}(j), Z_2^{(\ell)}(j), \ldots, Z_n^{(\ell)}(j) \right]$ as well as the list $\mathcal{I}^{(\ell)}$.
3. When a server S_j receives such a request, it generates a quasi-random polynomial $C_j^{(\ell)}$, of degree at most $k-1$, such that $C_j^{(\ell)}(0) = B_0(j)$. The server S_j distributes to each server S_i ($i \in \mathcal{I}^{(\ell)}$) the share $C_j^{(\ell)}(i)$. Each server S_i calculates

$$B_0^{(\ell)}(i) = \sum_{j \in \mathcal{I}^{(\ell)}} C_j^{(\ell)}(i) \times \prod_{\substack{d \in \mathcal{I}^{(\ell)} \\ d \neq i}} \frac{d}{d-i} \tag{7.1}$$

   and replies to $\mathcal{R}$ with the share $s_i^{(\ell)} = B_0^{(\ell)}(i) + \sum_{j=1}^{n} (\omega_j - \alpha) \times D_j^{(\ell)}(i)$.
4. After receiving k responses $s_i^{(\ell)}$ ($i \in \mathcal{I}^{(\ell)}$), $\mathcal{R}$ interpolates a univariate polynomial $R^{(\ell)}$ from the k points $\left[i, s_i^{(\ell)} \right]$ and calculates the chosen secret: $\omega_{e^{(\ell)}} = R^{(\ell)}(0)$.

Figure 7.2: Second Distributed Oblivious Transfer Protocol with Adaptive Queries (Unlimited Number of Queries)

with the same request, to respond with — possibly — different shares. To achieve this, it is proposed that the servers generate a new polynomial $Q^{(\ell)}$ for the run ℓ instead of reusing the same polynomial Q in each run as in Sect. D.

In the original protocol, it is observed that the polynomial B_0 generated by the sender can actually be considered as a sharing polynomial for the value ω_0. Consequently, applying the redistribution technique introduced by Desmedt and Jajodia [42], the servers contacted in run ℓ are able to calculate a new sharing polynomial, $B_0^{(\ell)}$, for ω_0. The polynomial $Q^{(\ell)}$

is then defined by:

$$Q^{(\ell)}(x, y_1, y_2, \ldots, y_{n-1}) = B_0^{(\ell)}(x) + \sum_{i=1}^{n-1} (\omega_i - \omega_0) \times y_i \,.$$

To avoid a specific processing of ω_0, the sparse polynomial generated by the sender is slightly transformed and the following modifications is introduced in Blundo et al.'s DOT protocol:

- In the set-up phase, the sender randomly selects an element $\alpha \in \mathbb{K}$. The quasi-random polynomial B_0 is such that $B_0(0) = \alpha$ and the coefficient of y_i is $\omega_i - \alpha$ $(1 \leq i \leq n)$.
- In run ℓ, in addition to the n values $D_1^{(\ell)}(j), D_2^{(\ell)}(j), \ldots, D_n^{(\ell)}(j)$ sent to the server S_j $(j \in \mathcal{I}^{(\ell)})$, the receiver sends the list $\mathcal{I}^{(\ell)}$ of contacted servers.
- Each contacted server S_j generates a quasi-random polynomial $C_j^{(\ell)}$, of degree at most $k-1$, such that $C_j^{(\ell)}(0) = B_0(j)$. Then S_j distributes $C_j^{(\ell)}(i)$ to S_i $(i \in \mathcal{I}^{(\ell)})$. With k shares $C_j^{(\ell)}(i)$, each server S_i calculates

$$B_0^{(\ell)}(i) = \sum_{j \in \mathcal{I}^{(\ell)}} C_j^{(\ell)}(i) \times \prod_{\substack{d \in \mathcal{I}^{(\ell)} \\ d \neq i}} \frac{d}{d-i} \,.$$

- The response $s_i^{(\ell)}$ returned by S_i to the receiver is then built on $B_0^{(\ell)}(i)$ instead of $B_0(i)$.

The resulting protocol is described in Fig. 7.2.

7.5.2 Security

The sender's security, the receiver's privacy against a coalition of servers and the sender's security against a coalition of the receiver and $k-1$ servers before the first run are the same as in the first protocol. Indeed, the redistribution protocol is secure and the receiver's contribution is exactly the same.

Now, it is shown that the protocol is correct.

At the end of run ℓ, the receiver has collected k shares $s_j^{(\ell)}$ $(j \in \mathcal{I}^{(\ell)})$ and interpolates

$$R^{(\ell)}(0) = \sum_{i \in \mathcal{I}^{(\ell)}} \left(s_i^{(\ell)} \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \right).$$

We show that $R^{(\ell)}(0) = \omega_{e^{(\ell)}}$.

$$R^{(\ell)}(0) = \sum_{i \in \mathcal{I}^{(\ell)}} \left(\left(B_0^{(\ell)}(i) + \sum_{j=1}^{n} (\omega_j - \alpha) \times D_j^{(\ell)}(i) \right) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \right)$$

$$= \underbrace{\sum_{i \in \mathcal{I}^{(\ell)}} \left(B_0^{(\ell)}(i) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \right)}_{=R_1} + \underbrace{\sum_{j=1}^{n} \left(\sum_{i \in \mathcal{I}_k^{(\ell)}} (\omega_j - \alpha) \times D_j^{(\ell)}(i) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \right)}_{=R_2}$$

If we replace $B_0^{(\ell)}$ with its value (7.1), we obtain

$$\begin{aligned}
R_1 &= \sum_{i \in \mathcal{I}^{(\ell)}} \left(\left(\sum_{j \in \mathcal{I}^{(\ell)}} C_j^{(\ell)}(i) \times \prod_{\substack{d \in \mathcal{I}^{(\ell)} \\ d \neq i}} \frac{d}{d-i} \right) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \right) \\
&= \sum_{j \in \mathcal{I}^{(\ell)}} \left(\left(\sum_{i \in \mathcal{I}^{(\ell)}} C_j^{(\ell)}(i) \times \prod_{\substack{d \in \mathcal{I}^{(\ell)} \\ d \neq i}} \frac{d}{d-i} \right) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \right) \\
&= \sum_{j \in \mathcal{I}^{(\ell)}} C_j^{(\ell)}(0) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \\
&= \sum_{j \in \mathcal{I}^{(\ell)}} B_0(j) \times \prod_{\substack{i \in \mathcal{I}^{(\ell)} \\ j \neq i}} \frac{j}{j-i} \\
&= B_0(0) = \alpha
\end{aligned}$$

In addition, we have

$$\begin{aligned}
R_2 &= \sum_{j=1}^{n} \left(\omega_j - \alpha \right) \times D_j^{(\ell)}(0) \\
&= \sum_{j=1}^{n} \left(\omega_j - \alpha \right) \delta_j^{e^{(\ell)}} \\
&= \omega_{e^{(\ell)}} - \alpha
\end{aligned}$$

It follows that $R^{(\ell)}(0) = R_1 + R_2 = \omega_{e^{(\ell)}}$. Therefore, the protocol is correct (security condition C_1' is satisfied).

7.6 Efficiency Consideration

In Table 7.1, the main computations performed by each party are listed for Blundo et al.'s DOT protocol repeated t times, and for the two DOT protocols proposed in this chapter.

Table 7.1: Computation Efficiency of Adaptive Distributed Oblivious Transfer Protocols

Party	**Blundo et al.'s DOT protocol**	**First Adaptive DOT protocol**	**Second Adaptive DOT protocol**
Set-up Phase(s)			
$\mathcal{S}$	2t sharing polynomials	2t sharing polynomials	2 sharing polynomials
Run ℓ $(1 \leq \ell \leq t)$			
$\mathcal{R}$	$n-1$ sharing polynomials, 2 interpolations, 1 division	n sharing polynomials, 2 interpolations, 1 division	n sharing polynomials, 2 interpolations, 1 division
S_i $(i \in \mathcal{I}^{(\ell)})$	2 $(n-1)$-tuple products	2 $(n-1)$-tuple products	2 sharing polynomials, 2 interpolations, 2 $(n-1)$-tuple products

The polynomial B_0 generated by the sender in the ℓ-th execution of Blundo et al.'s protocol is denoted by $B_0^{(\ell)}$ $(1 \leq \ell \leq t)$ and the transfer phase of the ℓ-th execution of Blundo et al.'s protocol is called run ℓ.

Similarly, Table 7.2 lists for Blundo et al.'s DOT protocol repeated t times, and for the two adaptive DOT protocols presented in this chapter the number of shares exchanged between the sender and the servers, the receiver and the servers, and amongst the servers.

To improve the fairness of the comparison, the masks of the sub-protocol described in [19, 20] are not taken into account, i.e., the $2 \times t \times m \times (n-1)$ shares distributed by the sender to the servers and the $2 \times t \times k$ shares collected by the receiver.

The computation and communication performance of the first protocol are close to the performance of Blundo et al.'s protocol executed t times (see Tables 7.1 and 7.2). The second protocol requires more computation on the servers' sides, but less computation on the sender's side, as shown in Table 7.1. In addition, the number of shares exchanged in the second protocol is slightly higher than the number of shares exchanged in the original protocol (see Table 7.2), due to the redistribution of the secret values α_1 and α_2 amongst servers.

Table 7.2: Communication Efficiency of Adaptive Distributed Oblivious Transfer Protocols

Party	**Blundo et al.'s DOT protocol**	**First Adaptive DOT protocol**	**Second Adaptive DOT protocol**
Set-up Phase(s)			
$\mathcal{S}$ to S_j $(1 \leq j \leq m)$	$2tn$ shares: $B_{1,0}^{(\ell)}(j)$, $B_{2,0}^{(\ell)}(j)$, $c_1\omega_1 - c_0\omega_0$, $c_2\omega_2 - c_0\omega_0$, $\ldots$, $c_{n-1}\omega_{n-1} - c_0\omega_0$, $c_1 - c_0$, $c_2 - c_0$, $\ldots$, $c_{n-1} - c_0$ $(1 \leq \ell \leq t)$	$2(n+t)$ shares: $B_{1,0}^{(1)}(j)$, $B_{1,0}^{(2)}(j)$, $\ldots$, $B_{1,0}^{(t)}(j)$, $B_{2,0}^{(1)}(j)$, $B_{2,0}^{(2)}(j)$, $\ldots$, $B_{2,0}^{(t)}(j)$, $c_1\omega_1 - \alpha_1$, $c_2\omega_2 - \alpha_1$, $\ldots$, $c_n\omega_n - \alpha_1$, $c_1 - \alpha_2$, $c_2 - \alpha_2$, $\ldots$, $c_n - \alpha_2$	$2(n+1)$ shares: $B_{1,0}(j)$, $B_{2,0}(j)$, $c_1\omega_1 - \alpha_1$, $\ldots$, $c_n\omega_n - \alpha_1$, $c_1 - \alpha_2$, $\ldots$, $c_n - \alpha_2$
Run ℓ $(1 \leq \ell \leq t)$			
$\mathcal{R}$ to S_i $(i \in \mathcal{I}^{(\ell)})$	$n-1$ shares: $D_1^{(\ell)}(i)$, $D_2^{(\ell)}(i)$, $\ldots$, $D_{n-1}^{(\ell)}(i)$	n shares: $D_1^{(\ell)}(i)$, $D_2^{(\ell)}(i)$, $\ldots$, $D_n^{(\ell)}(i)$	n shares: $D_1^{(\ell)}(i)$, $D_2^{(\ell)}(i)$, $\ldots$, $D_n^{(\ell)}(i)$
S_i to $\mathcal{R}$ $(i \in \mathcal{I}^{(\ell)})$	2 shares: $s_{1,i}^{(\ell)}$, $s_{1,i}^{(\ell)}$	2 shares: $s_{1,i}^{(\ell)}$, $s_{1,i}^{(\ell)}$	2 shares: $s_{1,i}^{(\ell)}$, $s_{1,i}^{(\ell)}$
S_i to S_j $(i \in \mathcal{I}^{(\ell)}$ and $j \in \mathcal{I}^{(\ell)})$	–	–	$2(k-1)$ shares: $C_{1,i}^{(\ell)}(j)$, $C_{2,i}^{(\ell)}(j)$

7.7 Conclusion

In this chapter were presented two polynomial interpolation-based unconditionally secure DOT protocols. Both protocols have the same levels of security and performance as Blundo et al.'s DOT protocol. In addition, they allow one or more receivers to adaptively obtain several of the sender's secrets.

In the next chapter is proposed the first information-theoretically secure threshold one-round DOT protocol, satisfying the four security conditions defined by Blundo et al. (see Sect. 1.3.2). This protocol allows a receiver to obtain a secret, provided she contacts at least k servers, where k is a fixed parameter of the protocol.

8

An Information-Theoretically Secure Threshold Distributed Oblivious Transfer Protocol

The unconditionally secure *distributed oblivious transfer* protocol presented by Blundo, D'Arco, De Santis, and Stinson at SAC 2002 [19] allows a receiver to contact k servers and obtain one out of n secrets held by a sender.

Once the protocol has been executed, the sender does not know which secret was selected by the receiver and the receiver knows nothing of the secrets she did not choose. In addition, the receiver's privacy is guaranteed against a coalition of $k-1$ servers and similarly, the sender's security is guaranteed against a coalition of $k-1$ servers. However, after the receiver has obtained a secret, she is able to learn all secrets by corrupting one server only. As a consequence, an external mechanism is required to prevent the receiver from contacting more than k servers.

The one-round distributed oblivious transfer protocol proposed in this chapter is unconditionally secure, allows the receiver to contact k servers or more, and guarantees the sender's security, even if the receiver corrupts $k-1$ servers after having obtained a secret.

8.1 Introduction

A *distributed oblivious transfer* (DOT) protocol enables a *receiver* to contact 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 some DOT protocols (e.g., [8, 19, 20, 29, 74, 78]), a restriction is observed in the definition above: the receiver is not allowed to contact "k or more" servers to obtain the chosen secret, but *exactly* k servers. If the receiver contacts more than k servers, the protocol's security is compromised. This is especially surprising for polynomial interpolation-based protocols which rely on Shamir's secret sharing scheme [89]. Indeed, Shamir's scheme allows a party to obtain a secret shared amongst m other parties, by contacting k or more of these parties. It may seem curious that this threshold property is not inherited by the DOT protocols themselves.

This chapter introduces the first information-theoretically secure threshold one-round DOT protocol. That is, the number of servers the receiver needs to contact to obtain a secret is not limited to k. Moreover, unlike other unconditionally secure DOT protocols, the proposed protocol satisfies security conditions C_1, C_2 for a coalition of any size (i.e., $\lambda_R = m$, where $m > 1$ is the number of servers participating in the protocol), C_3 for $\lambda_S = k - 1$, and C_4 for $\lambda_C = k - 1$ (security conditions and parameters are described in Sect. 2.2.3). Actually, Blundo, D'Arco, De Santis, and Stinson [19, 20] have proved that condition C_4 cannot be guaranteed with a one-round DOT protocol for $\lambda_C > 0$, a round being defined as a set of consistent requests/responses exchanged between the receiver and $t \geq k$ servers. To circumvent this limitation, the commodity-based model introduced by Beaver [5] is used. More precisely, the protocol is based on Rivest's trusted initializer oblivious transfer protocol [87]. In Rivest's protocol, an additional party — the *trusted initializer* — is involved in the set-up phase; he generates and distributes random values, but receives nothing from other parties (in particular, he obtains neither the sender's secrets, nor the receiver's choice). In addition, the proposed protocol has an efficiency similar to the efficiency of the full protocol presented by Blundo et al. [19, 20].

This chapter is organized as follows. In Sect. 8.2, an overview of the oblivious transfer protocol proposed by Rivest [87] is given. Section 8.3 introduces some definitions and notations, as well as the security model related to the proposed protocol. The protocol itself is described in Sect. 8.4 and its security is analysed in Sect. 8.5. The following section (Sect. 8.6) is devoted to the performance of the protocol and the last section (Sect. 8.7) concludes the chapter.

8.2 Background

The oblivious transfer protocol presented by Rivest [87] is based on the protocol introduced by Bennett, Brassard, Crépeau, and Skubiszewska [12], adapted to the trusted initializer model.

In Rivest's protocol, a sender $\mathcal{S}$ holds two secrets $\omega_0, \omega_1 \in [\![0, 1]\!]^\ell$ ($\ell \in \mathbb{N}, \ell > 0$) and a receiver $\mathcal{R}$ wishes to learn the secret ω_e ($e = 0$ or $e = 1$).

In the set-up phase, a trusted initializer $\mathcal{T}$ gives to $\mathcal{S}$ two random ℓ-bit strings r_0 and r_1. Then, $\mathcal{T}$ selects a random bit d and sends the pair $[\, d, r_d \,]$ to $\mathcal{R}$.

In the transfer phase, $\mathcal{R}$ selects the index e of one secret and transmits $c = e \oplus d$ to $\mathcal{S}$. The sender $\mathcal{S}$ replies with two values $f_0 = \omega_0 \oplus r_c$ and $f_1 = \omega_1 \oplus r_{1-c}$. To obtain ω_e, $\mathcal{R}$ calculates $f_e \oplus r_d$.

Clearly, the receiver obtains one secret only and the sender cannot determine which secret was chosen by the receiver.

8.3 Preliminaries

8.3.1 Notations and Definitions

The setting of the DOT protocol described in this chapter encompasses a sender $\mathcal{S}$ who owns n secrets $\omega_1, \omega_2, \ldots, \omega_n$ ($n > 1$) in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power), a receiver $\mathcal{R}$ who wishes to learn a secret ω_e ($1 \leq e \leq n$), a trusted initializer $\mathcal{T}$ who generates random elements of $\mathbb{K}$ and m servers S_i ($m > 1$ and $i \in \mathcal{I}_\mathrm{m} \subseteq \mathbb{K}^*$, where $|\mathcal{I}_\mathrm{m}| = m$). It is assumed that $p > \max(n, m)$ and that all operations are executed in $\mathbb{K}$. The set of indices of secrets $\omega_1, \omega_2, \ldots, \omega_n$ held by $\mathcal{S}$ is denoted by $\mathcal{I}_\mathrm{n} = [\![1, n]\!]$.

The protocol is composed of three phases: a *set-up phase*, a *commodity acquisition phase* and a *transfer phase*. In the set-up phase, for each secret the sender generates shares thanks to a (k, m)-threshold Shamir's secret sharing scheme [89] ($1 < k \leq m$). Then, the sender distributes the shares to the m servers and does not intervene in the rest of the protocol. In the commodity acquisition phase, the receiver contacts the trusted initializer who generates and distributes consistent masks to the m servers and to the receiver. The trusted initializer's presence is only required in this phase. In the transfer phase, the receiver has to contact t servers ($k \leq t \leq m$) to collect enough shares to construct ω_e.

The protocol requires the availability of unicast communication channels between the trusted initializer and the servers, between the trusted initializer and the receiver and between the sender and the servers. The receiver sends requests to the servers thanks to a broadcast channel and collects responses thanks to unicast channels between servers and herself. It is assumed that these unicast channels are secure, i.e., any party is unable to eavesdrop on them and that all channels guarantee that communications cannot be tampered with (see Sect. 2.2.1).

Since security conditions are linked to the quantity of information received by parties, it seems appropriate to use Shannon's entropy function [90, 91], and more generally

information theory, to demonstrate the security of the protocol (see Sect. 2.1.3 and for more details on information theory, see for example [33]).

8.3.2 Security Model

In this chapter, an active adversary (see Sect. 2.2.2) is assumed. This adversary wishes to obtain information he is not entitled to have, without aborting the protocol or providing the receiver with a forged secret. That is, all parties wish to complete the protocol to allow the receiver to obtain the chosen secret. In particular, the trusted initializer and the sender are honest. However, some servers may actively collaborate to determine the receiver's choice or the sender's secrets. The receiver may also actively cheat by corrupting servers, before or after the protocol is executed. In both cases, the receiver has of course access to all data held by the corrupted servers.

8.4 Protocol Description

The key idea underlying the DOT protocol presented in this chapter is to extend Rivest's OT protocol in two directions:

1. Generalization to n secrets
2. Introduction of a distributed model with m servers

Furthermore, to prevent the servers from learning the sender's secrets, they receive shares of the secrets held by the sender. These shares are generated thanks to Shamir's secret sharing schemes [89] (see Sect. 2.3.1).

In addition, to guarantee that the contacted servers do not receive requests related to different secrets, they all receive the same request broadcast by the receiver.

The full protocol is described in Fig. 8.1.

8.5 Security of the Protocol

8.5.1 Formal Model

To prove the security of the protocol, a formal model similar to the model introduced by Blundo et al. [19, 20] is used. In this model, it is assumed that the parties execute publicly known programs whose data are private. These data are described by the following discrete random variables shown on Fig. 8.2.

Let $\mathcal{I}_\mathrm{m}$ ($m > 1$) be a set of m distinct elements of $\mathbb{K}^*$ and S_j a server such that $j \in \mathcal{I}_\mathrm{m}$.

Input	The sender $\mathcal{S}$ contributes with n secrets $\omega_1, \omega_2, \ldots, \omega_n \in \mathbb{K}$ The trusted initializer $\mathcal{T}$ generates n sets of m random masks and randomly chooses one of the n sets The receiver $\mathcal{R}$ chooses an index $e \in \mathcal{I}_\mathrm{n} = [\![1, n]\!]$, and contributes with a cyclic permutation $\pi \in \mathfrak{S}_n$
Output	$\mathcal{R}$ receives ω_e, while $\mathcal{S}$ and $\mathcal{T}$ receive nothing.

Set-up Phase

1 - *Preparation of shares.* For each secret ω_i ($i \in \mathcal{I}_\mathrm{n}$), the sender $\mathcal{S}$ generates, thanks to a (k, m)-threshold Shamir's secret sharing scheme, a sharing polynomial F_i of $\mathbb{K}_{k-1}[X]$, such that $F_i(0) = \omega_i$.

2 - *Distribution of shares.* To each server S_j ($j \in \mathcal{I}_\mathrm{m}$), $\mathcal{S}$ transmits the n shares $F_1(j)$, $F_2(j), \ldots, F_n(j)$.

Commodity Acquisition Phase

1 - *Preparation of masks.* The trusted initializer $\mathcal{T}$ generates mn random masks $r_{j,i} \in \mathbb{K}$ ($j \in \mathcal{I}_\mathrm{m}$, $i \in \mathcal{I}_\mathrm{n}$) and one random index $s \in \mathcal{I}_\mathrm{n}$.

2 - *Distribution of masks.* $\mathcal{T}$ distributes the n masks $r_{j,1}, r_{j,2}, \ldots, r_{j,n}$ to the server S_j ($j \in \mathcal{I}_\mathrm{m}$) and the index s as well as the m masks $r_{j,s}$ to the receiver $\mathcal{R}$.

Transfer Phase

1 - *Selection of the secret index and generation of the corresponding request.* The receiver $\mathcal{R}$ chooses a secret index $e \in \mathcal{I}_\mathrm{n}$ and generates the cyclic permutation $\pi \in \mathfrak{S}_n$ which satisfies $\pi(e) = s$.

2 - *Selection of servers and broadcast of a query.* $\mathcal{R}$ selects a subset $\mathcal{I} \subseteq \mathcal{I}_\mathrm{m}$ of $t \geq k$ indices and broadcasts a query containing the first cyclic permutation item, $\pi(1)$, as well as the list $\mathcal{I}$.

3 - *Responses of the servers.* Each server S_ℓ such that $\ell \in \mathcal{I}$ returns $\mu_{\ell,i} = F_i(\ell) + r_{\ell,\pi(i)}$ ($i \in \mathcal{I}_\mathrm{n}$) to $\mathcal{R}$.

4 - *Construction of the requested secret.* For each of the t responses $\mu_{\ell,e}$, $\mathcal{R}$ calculates the share $\mu_{\ell,e} - r_{\ell,s} = F_e(\ell)$, interpolates F_e and obtains $\omega_e = F_e(0)$.

Figure 8.1: Threshold Distributed Oblivious Transfer Protocol – Overview

By extension, if $\boldsymbol{X}_j$ is a random variable which describes a datum x_j held by a server S_j ($j \in \mathcal{I}_\mathrm{m}$) and $G = [\, j_1, j_2, \ldots, j_t \,]$ is a list of t indices of $\mathcal{I}_\mathrm{m}$ ($t \in [\![1, m]\!]$), the random variable describing the sequence $\left[\, x_{j_1}, x_{j_2}, \ldots, x_{j_t} \,\right]$ is denoted by $\boldsymbol{X}_G = \left[\, \boldsymbol{X}_{j_1}, \boldsymbol{X}_{j_2}, \ldots, \boldsymbol{X}_{j_t} \,\right]$. By simplification, $\boldsymbol{X}_{\mathcal{I}_\mathrm{m}}$ is denoted by $\boldsymbol{X}$.

- Each secret $\omega_i \in \mathbb{K}$ ($i \in \mathcal{I}_\mathrm{n}$) is described by a random variable $\boldsymbol{W}^i$ and the sequence of secrets $\omega_1, \omega_2, \ldots, \omega_n$ by the random variable $\boldsymbol{W} = \left[\, \boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^n \,\right]$. Moreover,

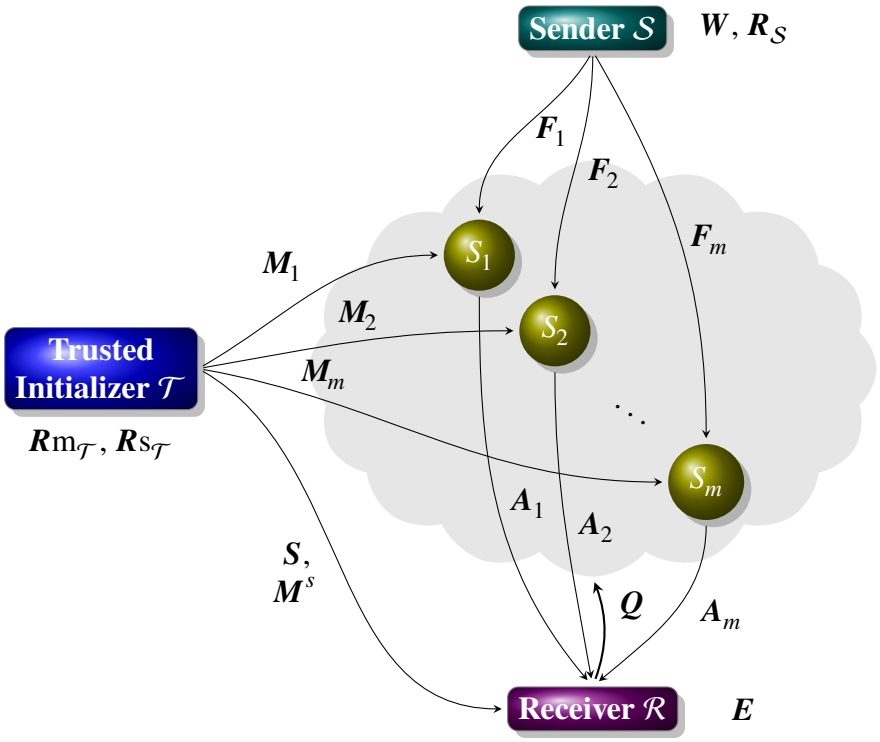

Figure 8.2: Threshold Distributed Oblivious Transfer Protocol – Random Variables

if $e \in \mathcal{I}_n$, the sequence $\left[\boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^{e-1}, \boldsymbol{W}^{e+1}, \boldsymbol{W}^{e+2}, \ldots, \boldsymbol{W}^n \right]$ is denoted by $\boldsymbol{W}^{\bar{e}}$.

- The secret index $e \in \mathcal{I}_n$ chosen by $\mathcal{R}$ is described by the random variable $\boldsymbol{E}$.
- The random variable $\boldsymbol{M}_j^i$ ($j \in \mathcal{I}_m$, $i \in \mathcal{I}_n$) corresponds to the mask $r_{j,i}$ and the random variable $\boldsymbol{M}_j$ ($j \in \mathcal{I}_m$) to the n ordered masks $\left[r_{j,1}, r_{j,2}, \ldots, r_{j,n} \right]$ distributed by $\mathcal{T}$ to the server S_j (i.e., $\boldsymbol{M}_j = \boldsymbol{M}_j^{\mathcal{I}_n} = \left[\boldsymbol{M}_j^1, \boldsymbol{M}_j^2, \ldots, \boldsymbol{M}_j^n \right]$). Similarly, the random variable $\boldsymbol{F}^i$ is associated with the polynomial $F_i \in \mathbb{K}_{k-1}[X]$ ($i \in \mathcal{I}_n$), the random variable $\boldsymbol{F}_j^i$ ($i \in \mathcal{I}_n, j \in \mathcal{I}_m$) with the share $F_i(j)$ and the random variable $\boldsymbol{F}_j$ ($j \in \mathcal{I}_m$) with the n shares $F_1(j), F_2(j), \ldots, F_n(j)$ distributed by $\mathcal{S}$ to the server S_j (i.e., $\boldsymbol{F}_j = \boldsymbol{F}_j^{\mathcal{I}_n} = \left[\boldsymbol{F}_j^1, \boldsymbol{F}_j^2, \ldots, \boldsymbol{F}_j^n \right]$). Note that $\mathbb{K}_{k-1}[X]$ is a finite field isomorphic to $\mathbb{K}^k$ since each polynomial $F \in \mathbb{K}_{k-1}[X]$ can be represented by an k-tuple of coefficients, elements of $\mathbb{K}$ (see Sect. 2.1.1.2). Therefore, the random variable $\boldsymbol{F}$ associated with the polynomial $F \in \mathbb{K}_{k-1}[X]$ is well defined.
- In addition, the random index $s \in \mathcal{I}_n$ chosen by $\mathcal{T}$ is described by the random variable $\boldsymbol{S}$. The notation $\boldsymbol{M}_j^s$ corresponds to the random variable describing $r_{j,s}$ and $\boldsymbol{M}^s$ is a shorthand for $\left[\boldsymbol{M}_{j_1}^s, \boldsymbol{M}_{j_2}^s, \ldots, \boldsymbol{M}_{j_m}^s \right]$ where $\mathcal{I}_m = \{ j_1, j_2, \ldots, j_m \}$.

- The cyclic permutation $\pi \in \mathfrak{S}_n$ is described by the random variable $\boldsymbol{Q}$:

$$H(\boldsymbol{Q} \mid \boldsymbol{E}, \boldsymbol{S}) = 0. \tag{8.1}$$

- The random variable $\boldsymbol{A}_j$ describes the answer $\boldsymbol{a}_j = \left[\mu_{j,1}, \mu_{j,2}, \ldots, \mu_{j,n} \right]$ returned by the server S_j ($j \in \mathcal{I}$) to the receiver on response to the broadcast request $q = \pi$. The random variable describing the answer $\mu_{j,i} = F_i(j) + r_{j,\pi(i)}$ ($j \in \mathcal{I}$, $i \in \mathcal{I}_\mathrm{n}$) is denoted by $\boldsymbol{A}_j^i$.

- A few uniform random variables are associated with the private data produced by the parties involved in the protocol:

  – The random masks $r_{j,i}$ ($i \in \mathcal{I}_\mathrm{n}$, $j \in \mathcal{I}_\mathrm{m}$) generated by the trusted initialiser $\mathcal{T}$ are described by the uniform random variable $\boldsymbol{R}\mathrm{m}_\mathcal{T}$. That is,

$$H(\boldsymbol{M}_j^i \mid \boldsymbol{R}\mathrm{m}_\mathcal{T}) = 0. \tag{8.2}$$

  Similarly, the secret index s generated by the trusted initialiser $\mathcal{T}$ is described by the uniform random variable $\boldsymbol{R}\mathrm{s}_\mathcal{T}$, i.e.,

$$H(\boldsymbol{S} \mid \boldsymbol{R}\mathrm{s}_\mathcal{T}) = 0. \tag{8.3}$$

  Note that since $H(\boldsymbol{M}^s \mid \boldsymbol{M}, \boldsymbol{S}) = 0$ then

$$H(\boldsymbol{M}^s \mid \boldsymbol{R}\mathrm{m}_\mathcal{T}, \boldsymbol{R}\mathrm{s}_\mathcal{T}) = 0. \tag{8.4}$$

  – The shares $F_i(j)$ ($i \in \mathcal{I}_\mathrm{n}$, $j \in \mathcal{I}_\mathrm{m}$) generated by the sender $\mathcal{S}$ are described by the uniform random variable $\boldsymbol{R}_\mathcal{S}$. Therefore,

$$H(\boldsymbol{F}_j^i \mid \boldsymbol{W}^i, \boldsymbol{R}_\mathcal{S}) = 0. \tag{8.5}$$

To show that security conditions C_1, C_2 for $\lambda_\mathrm{R} = m$, C_3 for $\lambda_\mathrm{S} = k - 1$, and C_4 for $\lambda_\mathrm{C} = k - 1$ are satisfied is equivalent to show properties listed in Table 8.1. The coalition controlled by an adversary is represented by the set G of indices of servers.

8.5.2 Correctness

Theorem 8.1. *The protocol described in Fig. 8.1 is correct (condition C_1 is satisfied for $t \in \mathbb{N}$ such that $k \leq t \leq m$), i.e., if all parties follow the protocol, the receiver obtains the chosen secret ω_e by contacting t servers S_j where $j \in \mathcal{I} = \{ j_1, j_2, \ldots, j_t \}$.*

Table 8.1: Security Conditions from an Information Theory Viewpoint

Security Condition	Number of Servers	Property
C_1	$k \leq \lvert\mathcal{I}\rvert \leq m$	$H(\boldsymbol{W}^e \mid \boldsymbol{E} = e, \boldsymbol{S}, \boldsymbol{M}^s, \boldsymbol{Q}, \boldsymbol{A}_{\mathcal{I}}) = 0$
C_2	$\lvert G\rvert \leq m$	$H(\boldsymbol{E} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{Q}) = H(\boldsymbol{E})$
C_3	$\lvert G\rvert \leq k-1$	$H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^s) = H(\boldsymbol{W})$
C_4	$k \leq \lvert\mathcal{I}\rvert \leq m$ $\lvert G\rvert \leq k-1$	$H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E} = e, \boldsymbol{S} = s, \boldsymbol{Q} = \pi, \boldsymbol{A}_{\mathcal{I}},$ $\boldsymbol{M}^s = \left[r_{1,s}, r_{2,s}, \ldots, r_{m,s} \right]) = H(\boldsymbol{W}^{\bar{e}})$

Proof.
Once $\mathcal{R}$ has chosen e, the cyclic permutation $\pi \in \mathfrak{S}_n$ such that $\pi(e) = s$ is determined. The response sent by the server S_ℓ ($\ell \in \mathcal{I}$) then contains the value $\mu_{\ell,e} = F_e(\ell) + r_{\ell,\pi(e)} = F_e(\ell) + r_{\ell,s}$. Since $\mathcal{R}$ has received $r_{\ell,s}$ from the trusted initializer in the commodity acquisition phase, she is able to calculate the t shares $F_e(\ell)$, to interpolate F_e (degree at most $k-1 < t$) and to determine $\omega_e = F_e(0)$.
It follows that $H(\boldsymbol{W}^e \mid \boldsymbol{E} = e, \boldsymbol{S}, \boldsymbol{M}^s, \boldsymbol{Q}, \boldsymbol{A}_{\mathcal{I}}) = 0$. □

8.5.3 Receiver's Privacy against a Coalition of Servers

Theorem 8.2. *The protocol described in Fig. 8.1 guarantees the receiver's privacy against a coalition of h servers S_j where $j \in G = \{ j_1, j_2, \ldots, j_h \}$ ($0 \leq h \leq m$), i.e., security condition C_2 with $\lambda_R = m$ is satisfied.*

Proof.
To show that $H(\boldsymbol{E} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{Q}) = H(\boldsymbol{E})$, first we demonstrate that

$$H(\boldsymbol{E} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{Q}) = H(\boldsymbol{E} \mid \boldsymbol{Q})$$

and second that

$$H(\boldsymbol{E} \mid \boldsymbol{Q}) = H(\boldsymbol{E}).$$

For the first part of the demonstration, we adapt a technique applied by Beimel, Chee, Wang, and Zhang [8] in a similar context.

1. First, we show that the conditional entropy of $\boldsymbol{E}$ given $\boldsymbol{F}_G$, $\boldsymbol{M}_G$ and $\boldsymbol{Q}$ satisfies

$$H(\boldsymbol{E} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{Q}) = H(\boldsymbol{E} \mid \boldsymbol{Q}).$$

For this purpose, thanks to property (2.8), we show that

$$H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{E}, \boldsymbol{Q}) = H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{Q}).$$

The choice of the receiver is independent from the data held by the trusted initializer, by the sender, by the servers and by herself at the end of the commodity acquisition phase, so

$$H(\boldsymbol{E} \mid \boldsymbol{R}\mathrm{m}_{\mathcal{T}}, \boldsymbol{R}\mathrm{s}_{\mathcal{T}}, \boldsymbol{W}, \boldsymbol{R}_{\mathcal{S}}, \boldsymbol{F}, \boldsymbol{M}, \boldsymbol{S}, \boldsymbol{M}^{s}) = H(\boldsymbol{E}). \tag{8.6}$$

If we apply property (2.10), we obtain the particular case

$$H(\boldsymbol{E} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{R}\mathrm{s}_{\mathcal{T}}) = H(\boldsymbol{E}). \tag{8.7}$$

Similarly, the uniform random variable $\boldsymbol{R}\mathrm{s}_{\mathcal{T}}$ held by the trusted initializer is independent from the uniform random variable $\boldsymbol{R}\mathrm{m}_{\mathcal{T}}$, from the sender's data and from the data held by the servers at the end of the commodity acquisition phase. It follows

$$H(\boldsymbol{R}\mathrm{s}_{\mathcal{T}} \mid \boldsymbol{R}\mathrm{m}_{\mathcal{T}}, \boldsymbol{W}, \boldsymbol{R}_{\mathcal{S}}, \boldsymbol{F}, \boldsymbol{M}) = H(\boldsymbol{R}\mathrm{s}_{\mathcal{T}}). \tag{8.8}$$

Once more, if we apply property (2.10), we obtain the particular case

$$H(\boldsymbol{R}\mathrm{s}_{\mathcal{T}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G) = H(\boldsymbol{R}\mathrm{s}_{\mathcal{T}}). \tag{8.9}$$

The joint entropy between $\boldsymbol{F}_G$ and $\boldsymbol{M}_G$ is

$$\begin{aligned}
H(\boldsymbol{F}_G, \boldsymbol{M}_G) &\geq H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{Q}) && \text{(from (2.3))} \\
&\geq H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{Q}, \boldsymbol{E}) && \text{(from (2.3))} \\
&\geq H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{R}\mathrm{s}_{\mathcal{T}}, \boldsymbol{E}) && \text{(from (8.1), (8.3) and (2.6))} \\
&= H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{R}\mathrm{s}_{\mathcal{T}}) && \text{(from (8.7) and (2.9))} \\
&= H(\boldsymbol{F}_G, \boldsymbol{M}_G). && \text{(from (8.9) and (2.9))}
\end{aligned}$$

Therefore, $H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{Q}) = H(\boldsymbol{F}_G, \boldsymbol{M}_G \mid \boldsymbol{Q}, \boldsymbol{E})$ and from property (2.8), $H(\boldsymbol{E} \mid \boldsymbol{Q}, \boldsymbol{F}_G, \boldsymbol{M}_G) = H(\boldsymbol{E} \mid \boldsymbol{Q})$.

2. To prove that $H(\boldsymbol{E} \mid \boldsymbol{Q}) = H(\boldsymbol{E})$, thanks to property (2.8), it is sufficient to show that $H(\boldsymbol{Q} \mid \boldsymbol{E}) = H(\boldsymbol{Q})$.

   First, we observe that given a secret index e and a cyclic permutation π, the random index s is uniquely determined: $s = \pi(e)$. Therefore, in terms of entropy, it follows that

$$H(\boldsymbol{S} \mid \boldsymbol{Q}, \boldsymbol{E}) = 0. \tag{8.10}$$

Second, the conditional joint entropy between $\boldsymbol{Q}$ and $\boldsymbol{S}$ given $\boldsymbol{E}$ is

$$\begin{aligned} H(\boldsymbol{Q},\boldsymbol{S} \mid \boldsymbol{E}) &= H(\boldsymbol{Q} \mid \boldsymbol{E}) + H(\boldsymbol{S} \mid \boldsymbol{Q},\boldsymbol{E}) && \text{(from (2.4))} \\ &= H(\boldsymbol{Q} \mid \boldsymbol{E}) && \text{(from (8.10))} \end{aligned}$$

and also

$$\begin{aligned} H(\boldsymbol{Q},\boldsymbol{S} \mid \boldsymbol{E}) &= H(\boldsymbol{S} \mid \boldsymbol{E}) + H(\boldsymbol{Q} \mid \boldsymbol{E},\boldsymbol{S}) && \text{(from (2.4))} \\ &= H(\boldsymbol{S} \mid \boldsymbol{E}) && \text{(from (8.1))} \end{aligned}$$

It follows that $H(\boldsymbol{Q} \mid \boldsymbol{E}) = H(\boldsymbol{S} \mid \boldsymbol{E})$.

If we apply property (2.10) to equality (8.6), we obtain the particular case $H(\boldsymbol{E} \mid \boldsymbol{S}) = H(\boldsymbol{E})$ which, combined with property (2.8) gives $H(\boldsymbol{S} \mid \boldsymbol{E}) = H(\boldsymbol{S})$. Therefore, $H(\boldsymbol{Q} \mid \boldsymbol{E}) = H(\boldsymbol{S})$. Moreover, because the random variable $\boldsymbol{S}$ is uniform, it holds $H(\boldsymbol{S}) = \log_2(n)$ and because the number of cyclic permutations of $\mathfrak{S}_n$ is n, we can write:

$$\log_2(n) \geq H(\boldsymbol{Q}) \geq H(\boldsymbol{Q} \mid \boldsymbol{E}) = H(\boldsymbol{S}) = \log_2(n).$$

It follows that $H(\boldsymbol{Q} \mid \boldsymbol{E}) = H(\boldsymbol{Q})$ and from (2.8) that $H(\boldsymbol{E} \mid \boldsymbol{Q}) = H(\boldsymbol{E})$.

We have shown that $H(\boldsymbol{E} \mid \boldsymbol{Q},\boldsymbol{F}_G,\boldsymbol{M}_G) = H(\boldsymbol{E} \mid \boldsymbol{Q})$ and $H(\boldsymbol{E} \mid \boldsymbol{Q}) = H(\boldsymbol{E})$. We conclude $H(\boldsymbol{E} \mid \boldsymbol{F}_G,\boldsymbol{M}_G,\boldsymbol{Q}) = H(\boldsymbol{E})$.

□

8.5.4 Sender's Security against a Coalition of the Receiver and Servers

Theorem 8.3. *The protocol described in Fig. 8.1 guarantees the sender's security against a coalition of the receiver and h servers S_j where $j \in G = \{ j_1, j_2, \ldots, j_h \}$ $(0 \leq h \leq k-1)$, before the protocol is executed (security condition C_3 with $\lambda_S = k-1$ is satisfied).*

Proof.
The demonstration is symmetrical to the previous demonstration. First, we demonstrate that $H(\boldsymbol{W} \mid \boldsymbol{F}_G,\boldsymbol{M}_G,\boldsymbol{E},\boldsymbol{S},\boldsymbol{M}^s) = H(\boldsymbol{W} \mid \boldsymbol{F}_G)$ and second that the secrets are independent from the shares received by any set of h servers $(h \leq k-1)$ in the set-up phase, i.e., $H(\boldsymbol{W} \mid \boldsymbol{F}_G) = H(\boldsymbol{W})$. These two demonstrations show that the secrets are independent from the data held by a coalition between the receiver and h servers.

1. The uniform random variable $\boldsymbol{R}\mathrm{m}_{\mathcal{T}}$ held by the trusted initializer is independent from the data held by the sender, by the servers and by herself at the end of the commodity acquisition phase, except masks. It follows

$$H(\boldsymbol{R}\mathrm{m}_{\mathcal{T}} \mid \boldsymbol{R}\mathrm{s}_{\mathcal{T}}, \boldsymbol{W}, \boldsymbol{R}_{\mathcal{S}}, \boldsymbol{F}) = H(\boldsymbol{R}\mathrm{m}_{\mathcal{T}}). \tag{8.11}$$

   If we apply property (2.10), we obtain the particular case

$$H(\boldsymbol{R}\mathrm{m}_{\mathcal{T}} \mid \boldsymbol{R}\mathrm{s}_{\mathcal{T}}, \boldsymbol{W}, \boldsymbol{F}_G) = H(\boldsymbol{R}\mathrm{m}_{\mathcal{T}}). \tag{8.12}$$

   Likewise, from (8.6) and (2.10) we can write

$$H(\boldsymbol{E} \mid \boldsymbol{W}, \boldsymbol{F}_G, \boldsymbol{R}\mathrm{s}_{\mathcal{T}}, \boldsymbol{R}\mathrm{m}_{\mathcal{T}}) = H(\boldsymbol{E}), \tag{8.13}$$

   and from (8.8) and (2.10) we can write

$$H(\boldsymbol{R}\mathrm{s}_{\mathcal{T}} \mid \boldsymbol{W}, \boldsymbol{F}_G) = H(\boldsymbol{R}\mathrm{s}_{\mathcal{T}}). \tag{8.14}$$

   The conditional entropy of $\boldsymbol{W}$ given $\boldsymbol{F}_G$ is

$$\begin{aligned} H(\boldsymbol{W} \mid \boldsymbol{F}_G) &\geq H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^s) && \text{(from (2.3))} \\ &\geq H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{E}, \boldsymbol{R}\mathrm{m}_{\mathcal{T}}, \boldsymbol{R}\mathrm{s}_{\mathcal{T}}) && \text{(from (8.2), (8.3), (8.4) and (2.6))} \\ &= H(\boldsymbol{W} \mid \boldsymbol{F}_G). && \text{(from (8.13), (8.14), (8.12) and (2.9))} \end{aligned}$$

   We conclude that $H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^s) = H(\boldsymbol{W} \mid \boldsymbol{F}_G)$.

2. It is well-known that Shamir's secret sharing scheme [89] is perfect, i.e, for $i \in \mathcal{I}_{\mathrm{n}}$, we have $H(\boldsymbol{W}^i \mid \boldsymbol{F}_G^i) = H(\boldsymbol{W}^i)$. The n secrets $\omega_1, \omega_2, \ldots, \omega_n$ are shared thanks to independent schemes; therefore, the previous equality may easily be generalized to the n-tuple of random variables $\left[\boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^n\right]$. It follows that $H(\boldsymbol{W} \mid \boldsymbol{F}_G^1, \boldsymbol{F}_G^2, \ldots, \boldsymbol{F}_G^n) = H(\boldsymbol{W})$.

   Since $\left[\boldsymbol{F}_G^1, \boldsymbol{F}_G^2, \ldots, \boldsymbol{F}_G^n\right] = \boldsymbol{F}_G^{\mathcal{I}_{\mathrm{n}}} = \boldsymbol{F}_G$, we obtain $H(\boldsymbol{W} \mid \boldsymbol{F}_G) = H(\boldsymbol{W})$.

We have demonstrated that $H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^s) = H(\boldsymbol{W} \mid \boldsymbol{F}_G)$ and that $H(\boldsymbol{W} \mid \boldsymbol{F}_G) = H(\boldsymbol{W})$. We conclude

$$H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^s) = H(\boldsymbol{W}).$$

□

8.5.5 Sender's Security against a 'Greedy' Receiver

Theorem 8.4. *The protocol described in Fig. 8.1 guarantees the sender's security against a coalition of the receiver and h servers S_ℓ where $\ell \in G = \{\ell_1, \ell_2, \ldots, \ell_h\}$ $(0 \leq h \leq k-1)$, after the protocol has been executed (security condition C_4 with $\lambda_C = k-1$ is satisfied).*

Proof.
We assume that in the transfer phase of the protocol, t servers S_j are contacted by the receiver, where $j \in \mathcal{I} = \{j_1, j_2, \ldots, j_t\}$ $(k \leq t \leq m)$. We introduce a random variable $\boldsymbol{K} = [\boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^s]$ describing the data $K = \left[e, s, \left[r_{1,s}, r_{2,s}, \ldots, r_{m,s}\right]\right]$. The theorem is demonstrated in four steps:

- First, we demonstrate that

$$\begin{aligned} &H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{A}_{\mathcal{I}}, \boldsymbol{K} = K, \boldsymbol{Q} = \pi) \\ &= H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{M}_G^{\bar{s}}, \boldsymbol{A}_{\mathcal{I}\setminus G}^{\bar{e}}, \boldsymbol{K} = K). \end{aligned}$$

- Second, we show that

$$\begin{aligned} &H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{M}_G^{\bar{s}}, \boldsymbol{A}_{\mathcal{I}\setminus G}^{\bar{e}}, \boldsymbol{K} = K) \\ &= H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{M}_G^{\bar{s}}, \boldsymbol{K} = K). \end{aligned}$$

- Third, we prove that

$$H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{M}_G^{\bar{s}}, \boldsymbol{K} = K) = H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K} = K).$$

- Lastly, we show that $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K} = K) = H(\boldsymbol{W}^{\bar{e}})$.

1. The random variable $\boldsymbol{A}_{\mathcal{I}}$ may be decomposed under the form $\boldsymbol{A}_{\mathcal{I}} = \left[\boldsymbol{A}_{\mathcal{I}\setminus G}, \boldsymbol{A}_G\right]$. Since $H(\boldsymbol{A}_G \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{Q} = \pi) = 0$, we apply property (2.7) which yields

$$\begin{aligned} &H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{A}_{\mathcal{I}}, \boldsymbol{K} = K, \boldsymbol{Q} = \pi) \\ &= H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{A}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K, \boldsymbol{Q} = \pi). \end{aligned}$$

   Similarly, the random variable $\boldsymbol{A}_{\mathcal{I}\setminus G}$ may be decomposed under the form $\boldsymbol{A}_{\mathcal{I}\setminus G} = \left[\boldsymbol{A}_{\mathcal{I}\setminus G}^{\bar{e}}, \boldsymbol{A}_{\mathcal{I}\setminus G}^{e}\right]$. Since for $j \in \mathcal{I}_{\mathrm{m}}$, we have $F_e(j) = (F_e(j) + r_{j,\pi(e)}) - r_{j,\pi(e)} = (F_e(j) + r_{j,\pi(e)}) - r_{j,s}$, it holds that

$$H(\boldsymbol{F}_{\mathcal{I}\setminus G}^{e} \mid \boldsymbol{A}_{\mathcal{I}\setminus G}^{e}, \boldsymbol{M}_{\mathcal{I}\setminus G}^{s}) = 0 \text{ and } H(\boldsymbol{A}_{\mathcal{I}\setminus G}^{e} \mid \boldsymbol{F}_{\mathcal{I}\setminus G}^{e}, \boldsymbol{M}_{\mathcal{I}\setminus G}^{s}) = 0.$$

Applying (2.7) we obtain

$$\begin{aligned} &H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{A}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K, \boldsymbol{Q} = \pi) \\ &= H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{F}^{e}_{\mathcal{I}\setminus G}, \boldsymbol{M}_G, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K, \boldsymbol{Q} = \pi). \end{aligned}$$

From properties (8.1) and (2.7), we obtain

$$\begin{aligned} &H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{F}^{e}_{\mathcal{I}\setminus G}, \boldsymbol{M}_G, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K, \boldsymbol{Q} = \pi) \\ &= H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{F}^{e}_{\mathcal{I}\setminus G}, \boldsymbol{M}_G, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K). \end{aligned}$$

Because $\boldsymbol{M}_G = \left[\, \boldsymbol{M}^{s}_G, \boldsymbol{M}^{\bar{s}}_G \,\right]$ and $\boldsymbol{F}_G = \left[\, \boldsymbol{F}^{e}_G, \boldsymbol{F}^{\bar{e}}_G \,\right]$, we can apply (2.7). It follows

$$\begin{aligned} &H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{F}^{e}_{\mathcal{I}\setminus G}, \boldsymbol{M}_G, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K) \\ &= H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}^{\bar{e}}_G, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K). \end{aligned}$$

2. To prove that

$$H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}^{\bar{e}}_G, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K) = H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}^{\bar{e}}_G, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K),$$

thanks to property (2.9) and to the fact that for three random variables $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ (respective alphabets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$) such that $\boldsymbol{X}$ and $[\,\boldsymbol{Y}, \boldsymbol{Z}\,]$ are independent, $H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = H(\boldsymbol{X}) = H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z} = z)$ for $z \in \mathcal{Z}$ (see Theorem E.4 in App. E.2), it is enough to show that

$$H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}_G, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K}, \boldsymbol{W}^{\bar{e}}) = H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}).$$

We have:

$$\begin{aligned} &H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) \\ &= H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) \\ &+ H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}, \boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G}) \\ &= H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) \\ &+ H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}). \end{aligned}$$

For $i \in \mathcal{I}_{\mathrm{n}}, i \neq e$ and $j \in \mathcal{I}_{\mathrm{m}}$, we have $F_i(j) = (F_i(j) + r_{j,\pi(i)}) - r_{j,\pi(i)}$. Using property (8.1), it holds that

$$H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_G, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}, \boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G}) = 0$$

and symmetrically

$$H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}) = 0.$$

It follows that

$$\begin{aligned} &H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) \\ &= H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}). \end{aligned}$$

Each mask $r_{j,i}$ ($i \in \mathcal{I}_{\mathrm{n}}$, $j \in \mathcal{I}_{\mathrm{m}}$) is randomly generated by the trusted initializer and is independent from the other variables held by the different parties at the beginning of the transfer phase. More precisely, if $G_1, G_2 \subseteq \mathcal{I}_{\mathrm{m}}$, and $H_1, H_2 \subseteq \mathcal{I}_{\mathrm{n}}$ are four subsets such that $G_1 \cap G_2 = \varnothing$ or $H_1 \cap H_2 = \varnothing$, we have

$$H(\boldsymbol{M}^{H_1}_{G_1} \mid \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{W}, \boldsymbol{F}, \boldsymbol{M}^{H_2}_{G_2}) = H(\boldsymbol{M}^{H_1}_{G_1}). \tag{8.15}$$

If we apply property (2.10) and thanks to Theorem E.4 (see App. E.2), we obtain the particular case

$$H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) = H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G}).$$

Therefore, $H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) = H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G})$.

Furthermore, the random variable $\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G}$ is uniform, so

$$H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G}) = \log_2 (p^{(n-1)\times|\mathcal{I}\setminus G|}).$$

Thus,

$$\begin{aligned} H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}) &\geq H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) && \text{(from (2.3))} \\ &= \log_2 (p^{(n-1)\times|\mathcal{I}\setminus G|}). \end{aligned}$$

By property (2.5), $H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}) \leq \log_2 (p^{(n-1)\times|\mathcal{I}\setminus G|})$.

It follows that $H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}) = \log_2 (p^{(n-1)\times|\mathcal{I}\setminus G|}) = H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G})$. We conclude

$$H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) = H(\boldsymbol{M}^{\bar{s}}_{\mathcal{I}\setminus G}) = H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}).$$

Applying property (2.10), we obtain $H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G} \mid \boldsymbol{F}^{\bar{e}}_{G}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K, \boldsymbol{W}^{\bar{e}}) = H(\boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G})$ and consequently

$$H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}^{\bar{e}}_{G}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{A}^{\bar{e}}_{\mathcal{I}\setminus G}, \boldsymbol{K} = K) = H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}^{\bar{e}}_{G}, \boldsymbol{F}^{e}_{\mathcal{I}}, \boldsymbol{M}^{\bar{s}}_{G}, \boldsymbol{K} = K).$$

3. Once again, we apply property (2.10) and Theorem E.4 to (8.15) and obtain the particular case

$$H(\boldsymbol{M}_G^{\bar{s}} \mid \boldsymbol{W}^{\bar{e}}, \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K} = K) = H(\boldsymbol{M}_G^{\bar{s}}).$$

It follows, from property (2.9), that

$$H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{M}_G^{\bar{s}}, \boldsymbol{K} = K) = H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K} = K).$$

4. Thanks to Lagrange's interpolation theorem, we can write $H(\boldsymbol{F}_{G'}^{e}, \boldsymbol{W}^{e} \mid \boldsymbol{F}_{\mathcal{I}}^{e}) = 0$ and $H(\boldsymbol{F}_{\mathcal{I}}^{e} \mid \boldsymbol{F}_{G'}^{e}, \boldsymbol{W}^{e}) = 0$ where G' is a set of $k-1$ distinct non-null indices. In particular, if $G' = G$ ($|G| = h < k$), we obtain $H(\boldsymbol{F}_G^{e}, \boldsymbol{W}^{e} \mid \boldsymbol{F}_{\mathcal{I}}^{e}) = 0$ and $H(\boldsymbol{F}_{\mathcal{I}}^{e} \mid \boldsymbol{F}_G^{e}, \boldsymbol{W}^{e}) = 0$. Using property (2.7), it follows that

$$H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K}) = H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{W}^{e}, \boldsymbol{K}).$$

In Sect. 8.5.4, we have demonstrated that if $|\mathcal{I}| < k$ then

$$H(\boldsymbol{W} \mid \boldsymbol{F}_{\mathcal{I}}, \boldsymbol{M}_{\mathcal{I}}, \boldsymbol{E}, \boldsymbol{S}, \boldsymbol{M}^{s}) = H(\boldsymbol{W}).$$

Applying this property to G and combining it with property (2.10) gives

$$H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{K}) = H(\boldsymbol{W}).$$

From property (2.8), $H(\boldsymbol{W} \mid \boldsymbol{F}_G, \boldsymbol{K}) = H(\boldsymbol{W})$ involves $H(\boldsymbol{F}_G, \boldsymbol{K} \mid \boldsymbol{W}) = H(\boldsymbol{F}_G, \boldsymbol{K})$ and from property (2.9), $H(\boldsymbol{F}_G, \boldsymbol{K} \mid \boldsymbol{W}) = H(\boldsymbol{F}_G, \boldsymbol{K} \mid \boldsymbol{W}^{e}, \boldsymbol{W}^{\bar{e}}) = H(\boldsymbol{F}_G, \boldsymbol{K})$ involves $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{K}, \boldsymbol{W}^{e}) = H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{W}^{e})$. We assume that the secrets are independent; consequently, $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{W}^{e}) = H(\boldsymbol{W}^{\bar{e}})$, which allows us to conclude $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{W}^{e}, \boldsymbol{K}) = H(\boldsymbol{W}^{\bar{e}})$, i.e., $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K}) = H(\boldsymbol{W}^{\bar{e}})$. Using Theorem E.4, it follows that $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G^{\bar{e}}, \boldsymbol{F}_{\mathcal{I}}^{e}, \boldsymbol{K} = K) = H(\boldsymbol{W}^{\bar{e}})$.

The demonstrations of the four steps above yield that $H(\boldsymbol{W}^{\bar{e}} \mid \boldsymbol{F}_G, \boldsymbol{M}_G, \boldsymbol{E} = e, \boldsymbol{S} = s, \boldsymbol{M}^{s} = \left[r_{1,s}, r_{1,s}, \ldots, r_{m,s}\right], \boldsymbol{Q} = \pi, \boldsymbol{A}_{\mathcal{I}}) = H(\boldsymbol{W}^{\bar{e}})$. □

8.6 Efficiency Consideration

Clearly, the number of shares returned by the servers to the receiver is higher with the proposed protocol (linear communication complexity in n) than with Beimel, Chee, Wang, and Zhang's DOT protocols [8] (sub-linear communication complexity in n for some private information retrieval protocols — see Sect. 1.2.2). However, in this section, it is

Table 8.2: Computation Efficiency of Distributed Oblivious Transfer Protocols

	Blundo et al.'s DOT Protocol	**Proposed DOT Protocol**
Set-up Phase		
$\mathcal{S}$	$2(n-1)$ random masks in $\mathbb{K}^*$, $2n$ sharing polynomials, $2mn$ shares	n sharing polynomials, mn shares
Commodity Acquisition Phase		
$\mathcal{T}$	–	mn random masks in $\mathbb{K}$, 1 random element of $\mathcal{I}_{\mathrm{n}}$
Transfer Phase		
$\mathcal{R}$	$(n-1)$ sharing polynomials, $k(n-1)$ shares, 4 polynomial interpolations	1 cyclic permutation of $\mathfrak{S}_n$, 1 polynomial interpolation
S_j $(j \in \mathcal{I})$	$2(n-1)$-tuple dot products, 2 additions	n additions

shown that the performance of Blundo et al.'s DOT protocol [19, 20] and of the protocol presented in this chapter are similar.

In Table 8.2 are listed the main computations performed by each party, for the proposed DOT protocol and for Blundo et al.'s DOT protocol.

Similarly, in Table 8.3 are listed for each protocol the number of shares exchanged between the sender and the servers, the receiver and the servers, and between the trusted initializer and (a) the sender and (b) the receiver in the case of the proposed protocol. It is assumed that in both protocols, k servers are contacted by the receiver, i.e., $t = k$ in the proposed protocol.

The operations performed off-line (set-up and commodity acquisition phases) for both protocols are close, but in the proposed protocol these operations are distributed between the sender and the trusted initializer. As for the on-line operations, the proposed protocol is more efficient than Blundo et al.'s one: on the receiver's side, only one cyclic permutation and one interpolation are required (vs. the generation of $k(n-1)$ shares from $(n-1)$ sharing polynomials and four interpolations in the case of Blundo et al.'s protocol), whereas on the servers' side, only n additions are required (vs. $2(n-1)$-tuple scalar products and two additions in the case of Blundo et al.'s protocol).

The number of shares distributed by the sender in the set-up phase is around $3n$ in Blundo et al.'s protocol and $2n$ in the proposed protocol. However, the proposed protocol

Table 8.3: Communication Efficiency of Distributed Oblivious Transfer Protocols (Shares)

	Blundo et al.'s DOT Protocol	**Proposed DOT Protocol**
Set-up Phase		
$\mathcal{S} \rightarrow S_j$ $(j \in \mathcal{I}_\mathrm{m})$	$2n$ shares, $n-1$ elements of $\mathbb{K}$	n shares
Commodity Acquisition Phase		
$\mathcal{T} \rightarrow S_j$ $(j \in \mathcal{I})$	–	n masks
$\mathcal{T} \rightarrow \mathcal{R}$	–	1 index, m masks
Transfer Phase		
$\mathcal{R} \rightarrow S_j$ $(j \in \mathcal{I})$	$n-1$ shares	$t = k$ server indices, 1 element of $\mathcal{I}_\mathrm{n}$ (nota: broadcast data)
$S_j \rightarrow \mathcal{R}$ $(j \in \mathcal{I})$	$2n$ shares	n shares

requires an additional distribution of $m(n+1)$ shares by the trusted initializer in the commodity acquisition phase. In the transfer phase, the request sent to a server contains $n-1$ shares (Blundo et al.'s protocol) whereas the broadcast request contains $k+1$ integers (the proposed protocol). The receiver collects two times more shares in Blundo et al.'s protocol than in the proposed protocol.

It is also worth pointing out that the DOT protocol presented in this chapter can easily be extended to a DOT-$\binom{n}{\ell}$; instead of choosing one set of random masks, the trusted initializer randomly selects ℓ sets of random masks and distributes them to the receiver in the commodity acquisition phase, with the corresponding indices $s_1, s_2, \ldots, s_\ell$. In this scenario, the receiver selects ℓ indices $e_1, e_2, \ldots, e_\ell$ and generates a random permutation π, instead of a cyclic permutation, such that $\pi(e_1) = s_1, \pi(e_2) = s_2, \ldots, \pi(e_\ell) = s_\ell$. The operations executed by the servers are the same as in the case where the receiver wishes to obtain one secret only. On reception of the responses, the receiver has to interpolate ℓ polynomials to determine the ℓ chosen secrets. Therefore, in the proposed protocol, due to the constant number of operations performed by the servers and to the constant number of data exchanged between the servers and the receiver, the communication and computation performance, relative to ℓ, improves when ℓ increases. Blundo et al.'s DOT protocol would need to be executed ℓ times for ℓ secrets, which would be less efficient than the proposed protocol.

In a similar vein, the protocol may easily be extended to a verifiable DOT protocol, with the simple requirement that enough shares are collected by the receiver to identify — and discard — incorrect shares returned by malicious servers. Thus, a Reed-Solomon codes [85] decoding algorithm like the algorithm introduced by Gao [52] would allow the receiver to determine the chosen secret in spite of $u \leq \frac{t-k}{2}$ malicious servers.

8.7 Conclusion

In this chapter was proposed the first strongly private (see Sect. 1.3.2) information-theoretically secure threshold one-round DOT protocol. This protocol allows a receiver to obtain a secret shared amongst m other parties, by contacting k or more of these parties.

The next chapter concludes this thesis. The results presented in this document are summarised and directions for possible future work are given.

9

Conclusion

This brief chapter presents a summary of results and outlines some directions for future research.

One of the contributions described in this thesis is the critical analysis of the few *distributed oblivious transfer* (DOT) protocols designed in years 2000s, showing the flaws and weaknesses of some of them. Also, five novel variants of DOT protocols were presented, either with a better efficiency than the existing protocols, or with additional features. More precisely, the following DOT protocols were proposed:

- a strongly secure DOT protocol (Chap. 4),
- a verifiable DOT protocol (Chap. 5),
- a t-out-of-n DOT protocol (Chap. 6),
- two adaptive DOT protocols (Chap. 7), and
- a threshold DOT protocol (Chap. 8).

Incidentally, the perfectness of Shamir's secret sharing scheme was demonstrated, using an information-theoretic model, and a new scheme was designed to prevent, in a DOT protocol, a receiver from obtaining a linear combination of secrets (with more than a non-null coefficient). While these results contribute significantly to the state of the art, there are still several interesting research directions to be explored to design a DOT protocol solution to the initial concern outlined in Chap. 1, namely, how to protect at the same time the privacy of Internet digital goods' buyers and sellers. Indeed, if a malicious receiver is assumed, all existing DOT protocols including the protocol variants presented in this thesis, either are not strongly secure (that is, they do not satisfy Blundo, D'Arco, De Santis,

and Stinson's four security conditions B_1, B_2, B_3, and B_4 [20], defined in Sect. 1.3.2), or are not efficient enough in terms of communication. The most obvious directions to explore are given by the limitations presented in Sect. 1.4:

- Efficient computationally secure DOT protocols is the first area to investigate. A simple question is: is it possible to design a DOT protocol much more efficient and secure than the parallel execution of oblivious transfer protocols by a receiver and k servers?
- Even though quantum oblivious transfer protocols have been heavily studied [37], the same question has not been tackled for computational security: is it possible to design an efficient quantum DOT protocol?
- Only a few variants of DOT protocols were designed and presented in this thesis; when secret sharing schemes variants are looked at, much remains to be done. A well-known strategy to design novel DOT protocols is to relax some parameters (for example, a perfect correctness may not be necessary) or to modify the DOT protocols' settings. Thus, DOT protocols relying on specific communication features (noisy channel, time-delay, packet reordering, etc.) could be investigated.

Finally, even if there is much research left to be done, current DOT protocols could have practical applications in some specific scenarios, in particular for a small number of random secrets held by a sender. This could be the case, for example, for encryption or decryption keys.

Appendix A

Demonstration of the Security of Shamir's Secret Sharing Scheme

In a k-threshold secret sharing scheme, a dealer who holds a secret s distributes parts of this secret (the *shares*) to n players; if k or more of these players pool their shares, they are able to determine s, but if less than k of them pool their shares, they cannot infer any information on s. In 1979, Shamir introduced such a scheme [89], based on polynomial interpolation in a finite field. This scheme is widely used in other cryptographic protocols, because it is simple, elegant and above all information-theoretically secure. There are a few proofs of the scheme's security, but to my knowledge, none of them is entirely based on the information-theoretic entropy function introduced by Shannon in 1948 [90, 91]. Such a demonstration is proposed below.

A.1 Introduction

Threshold secret sharing schemes were introduced by Blakley [18] and Shamir [89] in 1979. These schemes encompass a dealer and n players $P_1, P_2, \ldots, P_n$ ($n > 1$). The dealer holds a secret s from which he generates shares that he privately distributes to the players. Assuming that the threshold of a scheme is k ($1 < k \leq n$) and that the security model of this scheme is semi-honest (see Sect. 2.2.2), t players ($k \leq t \leq n$) pool their shares, which allows them to determine the secret s.

Shamir's scheme relies on the unisolvence theorem which states that given d points $[\, x_i, y_i \,]$ ($1 \leq i \leq d$) in a finite field $\mathbb{K}^2$ where $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power),

$0 < d \leq p$ and abscissae x_i are distinct, there exists a unique polynomial F of $\mathbb{K}_{d-1}[X]$ such that for $i = 1, 2, \ldots, d$, the equality $y_i = F(x_i)$ is satisfied.

Practically, in Shamir's secret sharing scheme, the dealer secretly chooses $k - 1$ random elements $q_1, q_2, \ldots, q_{k-1}$ of $\mathbb{K}$ and forms the polynomial

$$F = \sum_{\ell=1}^{k-1} q_\ell X^\ell + s\,.$$

Then, the dealer gives (in private) the share $F(i)$ to the player P_i ($i \in \mathcal{I}$, where $\mathcal{I} \subseteq \mathbb{K}^*$ and $1 < |\mathcal{I}| < p$).

In the secret reconstruction phase, t players P_j ($j \in \mathcal{J} \subseteq \mathcal{I}$, $|\mathcal{J}| = t \geq k$) pool their shares $F(j)$ and apply Lagrange's interpolation formula

$$s = \sum_{j \in \mathcal{J}} F(j) \prod_{\substack{i \in \mathcal{J} \\ i \neq j}} \frac{i}{i - j}$$

to recover the secret s.

Shamir's scheme is the key building block in numerous areas of threshold cryptographic protocols like key distribution protocols (e.g. [94]), key agreement protocols (e.g. [71]), Byzantine agreement protocols (e.g. [82]), multi-party computation protocols (e.g. [14, 27]), digital signature protocols (e.g. [41]), etc. (for more details on secret sharing schemes, see for example [7, 93]).

This broad usage is due to the simplicity of the scheme and to its security. Indeed, Shamir's scheme enjoys two basic security properties: *correctness* and *perfectness*.

- The *correctness* property means that at the end of the reconstruction phase, the secret s is determined with certainty. The correctness of Shamir's scheme is easy to demonstrate. Indeed, from their t pooled shares, the players involved in the reconstruction phase are able to build a system of k linear equations in k unknowns (the k coefficients of the sharing polynomial F). The matrix of the system is a Vandermonde matrix, which is non singular. By consequence, the system has a unique solution over $\mathbb{K}$. This unique solution allows the players to completely determine the sharing polynomial, in particular its free coefficient, i.e., the dealer's secret s.

- The *perfectness* property of the scheme means that a set of less than k players pooling their shares does not have any information on the dealer's secret. The most common proof of the perfectness of Shamir's scheme does not make any assumption on the probability distribution of the secret s (for example [89, 93]). The proof relies on

the fact that, given any set of $k-1$ shares, any k-th share corresponds to a different secret s. Recently, a demonstration assuming a probability distribution of s was proposed by Beimel [7], who showed that for any secret $s \in \mathbb{K}$, the probability for any sequence of $k-1$ shares is $1/|\mathbb{K}|^{k-1}$.

To my knowledge, there has never been a formal information-theoretic demonstration of the security of Shamir's secret sharing scheme, uniquely based on Shannon's entropy inequalities [90, 91]. In this chapter, such a demonstration is given. Section A.2 introduces some notations and definitions as well as the security model employed. The scheme's security itself is analysed in Sect. A.3.

A.2 Preliminaries

A.2.1 Notations and Definitions

Operations are executed in a finite field $\mathbb{K} = \mathrm{GF}(p)$ where $p \in \mathbb{N}$ is a prime power and $1 < n < p$.

The scheme requires the availability of private communication channels between the dealer and the players, and amongst the players who pool their shares. It is assumed that private channels are secure, i.e., no party is able to eavesdrop on them and that all channels guarantee that communications cannot be tampered with.

A.2.2 Security Model

It is assumed that the dealer is honest and that the players are honest but curious; in particular, some players may try to obtain the secret s by pooling less that k shares.

A.2.3 Theory of Information

Since security conditions are linked to the quantity of information received by parties, it seems appropriate to use Shannon's entropy function [90, 91], and more generally information theory, to demonstrate the security of Shamir's scheme. The definitions and properties given in Sect. 2.1.3 will be used (for more details on information theory, see for example [33]).

A.3 Security of Shamir's Secret Sharing Scheme

For a threshold k ($1 < k \leq n$), the sharing polynomial $F \in \mathbb{K}_{k-1}[X]$ generated by the dealer is

$$F = q_0 + q_1 X + \cdots + q_{k-1} X^{k-1} ,$$

where $q_0 = s$ and the coefficients q_i ($1 \leq i \leq k-1$) are randomly selected in $\mathbb{K}$. It is assumed that t players $P_{i_1}, P_{i_2}, \ldots, P_{i_t}$ ($i_1, i_2, \ldots, i_t \in \mathcal{J} \subseteq \mathbb{K}^*$) pool their t shares $F(i_1)$, $F(i_2), \ldots, F(i_t)$. The random variable describing the secret s is denoted by $\boldsymbol{S}$ and the random variable describing the share $F(i)$ ($i \in \mathcal{I} \subseteq \mathbb{K}^*$) is denoted by $\boldsymbol{F}_i$. The random variable describing the polynomial coefficient q_ℓ ($0 \leq \ell \leq k-1$) is denoted by $\boldsymbol{Q}_\ell$.

To demonstrate that Shamir's secret sharing scheme is correct, i.e., that any set of k or more players pooling their shares is able to determine the secret s, it is shown that for $t \geq k$ the entropy of the secret, knowing t shares, is null, that is:

$$H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_t}) = 0 . \tag{A.1}$$

Moreover, to demonstrate that the scheme is perfect, i.e., that any set of less than k players pooling their shares does not have any information on s, it is shown that for $t \leq k-1$ the entropy of the secret, knowing t shares, is the entropy of the secret, that is:

$$H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_t}) = H(\boldsymbol{S}) . \tag{A.2}$$

A.3.1 Correctness

Theorem A.1. *Shamir's secret sharing scheme is* correct.

Proof.
According to the unisolvence theorem, given k points $[\, a_1, b_1 \,], [\, a_2, b_2 \,], \ldots, [\, a_k, b_k \,]$ of $\mathbb{K}^2$, where abscissae a_i are distinct, there exists a unique polynomial

$$F = q_0 + q_1 X + \cdots + q_{k-1} X^{k-1}$$

of $\mathbb{K}_{k-1}[X]$ such that $F(a_i) = b_i$ for $i = 1, 2, \ldots, k$. Thus, in Shamir's scheme, the sharing polynomial F — that is, the k coefficients $q_0, q_1, \ldots, q_{k-1}$ — is uniquely determined by the k pairs $[\, i_1, F(i_1) \,], [\, i_2, F(i_2) \,], \ldots, [\, i_k, F(i_k) \,]$. From an information theory point of view, it follows that

$$H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_k}) = 0 . \tag{A.3}$$

Furthermore, the polynomial F, and by consequence the shares $F(i)$ $(i = i_1, i_2, \ldots, i_k)$, are completely determined by the knowledge of the coefficients $q_0, q_1, \ldots, q_{k-1}$. From an information theory viewpoint,

$$H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_k} \mid \boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}) = 0\,. \tag{A.4}$$

From equalities (A.3), (A.4) and property (2.6), we can write

$$H(\boldsymbol{S} \mid \boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}) = H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_k})\,.$$

In addition, $q_0 = s$. Therefore,

$$H(\boldsymbol{S} \mid \boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}) = H(\boldsymbol{S} \mid \boldsymbol{S}, \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) = 0\,.$$

Hence, $H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_k}) = 0$.
Using property (2.3), it holds

$$H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_t}) \leq H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_k}) = 0\,.$$

By property (2.5), it follows that

$$H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_t}) = 0\,.$$

This demonstrates that the scheme is correct. □

A.3.2 Perfectness

Theorem A.2. *Shamir's secret sharing scheme is* perfect.

Proof.
The joint entropy between the k-tuples $[\,\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}\,]$ and $\left[\,\boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}\,\right]$ is

$$\begin{aligned}
&H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}, \boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \\
&= H(\boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) + H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1} \mid \boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(from (2.4))} \\
&= H(\boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}})\,. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(from (A.3))}
\end{aligned}$$

Similarly,

$$\begin{aligned}
&H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}, \boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \\
&= H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}) + H(\boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}} \mid \boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}) \quad \text{(from (2.4))} \\
&= H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1})\,. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(from (A.4))}
\end{aligned}$$

Therefore,

$$H(\boldsymbol{F}_0, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) = H(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{k-1}),$$

i.e., because $F(0) = q_0 = s$,

$$H(\boldsymbol{S}, \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) = H(\boldsymbol{S}, \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}). \tag{A.5}$$

Using property (2.4), we develop the two members of (A.5):

$$\begin{aligned} &H(\boldsymbol{S}) + H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}} \mid \boldsymbol{S}) \\ &= H(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) + H(\boldsymbol{S} \mid \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}). \end{aligned}$$

The coefficients $q_1, q_2, \ldots, q_{k-1}$ are randomly chosen in $\mathbb{K}$. Consequently, we have

$$H(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) = \log_2 |\mathbb{K}|^{k-1}.$$

Moreover, the coefficients $q_1, q_2, \ldots, q_{k-1}$ are independent from the secret s, that is,

$$H(\boldsymbol{S} \mid \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) = H(\boldsymbol{S}).$$

It follows

$$H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}} \mid \boldsymbol{S}) = \log_2 |\mathbb{K}|^{k-1}.$$

From property (2.3), we can write

$$H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \geq H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}} \mid \boldsymbol{S}),$$

so $H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \geq \log_2 |\mathbb{K}|^{k-1}$. By property (2.5), we have

$$H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \leq \log_2 |\mathbb{K}|^{k-1},$$

so we conclude $H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) = \log_2 |\mathbb{K}|^{k-1}$. Using property (2.4), we develop again the two members of (A.5):

$$\begin{aligned} &H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) + H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \\ &= H(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) + H(\boldsymbol{S} \mid \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}). \end{aligned}$$

Since

$$H(\boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) = H(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) = \log_2 |\mathbb{K}|^{k-1}$$

and

$$H(\boldsymbol{S} \mid \boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_{k-1}) = H(\boldsymbol{S}),$$

we obtain

$$H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) = H(\boldsymbol{S}).$$

Because $t \leq k-1$, by property (2.3) we have

$$\begin{aligned} H(\boldsymbol{S}) &\geq H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_t}) \\ &\geq H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_{k-1}}) \\ &= H(\boldsymbol{S})\,. \end{aligned}$$

It follows that

$$H(\boldsymbol{S} \mid \boldsymbol{F}_{i_1}, \boldsymbol{F}_{i_2}, \ldots, \boldsymbol{F}_{i_t}) = H(\boldsymbol{S})\,.$$

Consequently the scheme is perfect. □

Appendix B

Conditional Entropies of Secrets Given a Linear Combination of these Secrets

Let x and y be two secrets selected in a finite field $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime power) and $\boldsymbol{X}$ and $\boldsymbol{Y}$ the corresponding random variables. The joint probability mass function of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is denoted by $p_{\boldsymbol{XY}}$. It is assumed that a party learns the linear combination $z = ax + by$ (z, a and b being elements of $\mathbb{K}$), i.e., the party learns z, a and b. The random variable associated with z is denoted by $\boldsymbol{Z}$. It is shown that in this scenario, the party may obtain information on x and y. Formally, it is sufficient to find a joint probability mass function $p_{\boldsymbol{XY}}$ such that the entropies of the secrets and the conditional entropies of the secrets given the linear combination satisfy the inequalities

$$\begin{cases} H(\boldsymbol{X} \mid \boldsymbol{Z}) < H(\boldsymbol{X}) \text{ and} \\ H(\boldsymbol{Y} \mid \boldsymbol{Z}) < H(\boldsymbol{Y}). \end{cases}$$

In the following sections, such a probability mass function is exhibited in two examples, both for dependent and independent random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$.

B.1 Dependent Random Variables

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two discrete random variables with alphabet $\mathbb{K} = \mathrm{GF}(2)$ and let $p_{\boldsymbol{XY}}$ be their joint probability mass function defined by

$$\begin{cases} p_{\boldsymbol{XY}}(0,0) = 1/32 \\ p_{\boldsymbol{XY}}(0,1) = 4/32 \\ p_{\boldsymbol{XY}}(1,0) = 16/32 \\ p_{\boldsymbol{XY}}(1,1) = 11/32 \end{cases}$$

The marginal probability mass functions $p_{\boldsymbol{X}}$ and $p_{\boldsymbol{Y}}$ are defined by:

$$\begin{cases} p_{\boldsymbol{X}}(0) = 1/32 + 4/32 = 5/32 \approx 0.16 \\ p_{\boldsymbol{X}}(1) = 16/32 + 11/32 = 27/32 \approx 0.84 \\ p_{\boldsymbol{Y}}(0) = 1/32 + 16/32 = 17/32 \approx 0.53 \\ p_{\boldsymbol{Y}}(1) = 4/32 + 11/32 = 15/32 \approx 0.47 \end{cases}$$

The corresponding entropies are then:

$$\begin{cases} H(\boldsymbol{X}) = -\sum_{x \in \mathbb{K}} p_{\boldsymbol{X}}(x) \log_2(p_{\boldsymbol{X}}(x)) \approx 0.63 & \text{(B.1)} \\ H(\boldsymbol{Y}) = -\sum_{y \in \mathbb{K}} p_{\boldsymbol{Y}}(y) \log_2(p_{\boldsymbol{Y}}(y)) \approx 1.00 & \text{(B.2)} \end{cases}$$

It is assumed that a party obtains the pair $[\, a, b \,] \in \mathbb{K}^2$ as well as the linear combination $z = ax + by$ (the random variable associated with the linear combination z is denoted by $\boldsymbol{Z}$). For example, $a = 1$, $b = 1$ and $z = 0$. Since $z = 0$ (i.e., $p_{\boldsymbol{Z}}(0) = 1$ and $p_{\boldsymbol{Z}}(1) = 0$), either $[\, x, y \,] = [\, 0, 0 \,]$ or $[\, x, y \,] = [\, 1, 1 \,]$. It follows:

$$\begin{cases} p_{\boldsymbol{XY}|\boldsymbol{Z}}(0, 0 \mid 0) = 1/(1 + 11) \approx 0.08 \\ p_{\boldsymbol{XY}|\boldsymbol{Z}}(1, 1 \mid 0) = 11/(1 + 11) \approx 0.92 \end{cases}$$

In addition,

$$\begin{cases} p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0) = p_{\boldsymbol{Y}|\boldsymbol{Z}}(0 \mid 0) = p_{\boldsymbol{XY}|\boldsymbol{Z}}(0, 0 \mid 0) \approx 0.08 \\ p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0) = p_{\boldsymbol{Y}|\boldsymbol{Z}}(1 \mid 0) = p_{\boldsymbol{XY}|\boldsymbol{Z}}(1, 1 \mid 0) \approx 0.92 \end{cases}$$

The conditional entropy $H(\boldsymbol{X} \mid \boldsymbol{Z})$ is

$$\begin{aligned} H(\boldsymbol{X} \mid \boldsymbol{Z}) &= p_{\boldsymbol{Z}}(0) H(\boldsymbol{X} \mid \boldsymbol{Z} = 0) + p_{\boldsymbol{Z}}(1) H(\boldsymbol{X} \mid \boldsymbol{Z} = 1) \\ &= H(\boldsymbol{X} \mid \boldsymbol{Z} = 0) \\ &= -p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0) \log_2(p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0)) - p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0) \log_2(p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0)) \\ &\approx 0.41 \end{aligned} \qquad \text{(B.3)}$$

Similarly,

$$H(\boldsymbol{Y} \mid \boldsymbol{Z}) \approx 0.41. \qquad \text{(B.4)}$$

From (B.1) and (B.3), it is concluded that $H(\boldsymbol{X} \mid \boldsymbol{Z}) < H(\boldsymbol{X})$ and from (B.2) and (B.4), the inequality $H(\boldsymbol{Y} \mid \boldsymbol{Z}) < H(\boldsymbol{Y})$ is obtained.

Note that neither $H(\boldsymbol{X} \mid \boldsymbol{Y}) = 0$ nor $H(\boldsymbol{Y} \mid \boldsymbol{X}) = 0$. (This is an assumption in Blundo, D'Arco, De Santis, and Stinson's DOT protocol [20].) Indeed,

$$\begin{aligned} H(\boldsymbol{X} \mid \boldsymbol{Y}) &= p_{\boldsymbol{Y}}(0)H(\boldsymbol{X} \mid \boldsymbol{Y} = 0) + p_{\boldsymbol{Y}}(1)H(\boldsymbol{X} \mid \boldsymbol{Y} = 1) \\ &= -p_{\boldsymbol{Y}}(0)p_{\boldsymbol{X}|\boldsymbol{Y}}(0 \mid 0)\log_2(p_{\boldsymbol{X}|\boldsymbol{Y}}(0 \mid 0)) \\ &\quad - p_{\boldsymbol{Y}}(0)p_{\boldsymbol{X}|\boldsymbol{Y}}(1 \mid 0)\log_2(p_{\boldsymbol{X}|\boldsymbol{Y}}(1 \mid 0)) \\ &\quad - p_{\boldsymbol{Y}}(1)p_{\boldsymbol{X}|\boldsymbol{Y}}(0 \mid 1)\log_2(p_{\boldsymbol{X}|\boldsymbol{Y}}(0 \mid 1)) \\ &\quad - p_{\boldsymbol{Y}}(1)p_{\boldsymbol{X}|\boldsymbol{Y}}(1 \mid 1)\log_2(p_{\boldsymbol{X}|\boldsymbol{Y}}(1 \mid 1)) \end{aligned}$$

From the definition of the joint probability mass function $p_{\boldsymbol{XY}}$, it is possible to write

$$\begin{cases} p_{\boldsymbol{X}|\boldsymbol{Y}}(0 \mid 0) = 1/(1+16) \approx 0.06 \\ p_{\boldsymbol{X}|\boldsymbol{Y}}(1 \mid 0) = 16/(1+16) \approx 0.94 \\ p_{\boldsymbol{X}|\boldsymbol{Y}}(0 \mid 1) = 4/(4+11) \approx 0.27 \\ p_{\boldsymbol{X}|\boldsymbol{Y}}(1 \mid 1) = 11/(4+11) \approx 0.73 \end{cases}$$

Thus,

$$\begin{aligned} H(\boldsymbol{X} \mid \boldsymbol{Y}) &\approx -0.53(0.06\log_2(0.06)) \\ &\quad - 0.53(0.94\log_2(0.94)) \\ &\quad - 0.47(0.27\log_2(0.27)) \\ &\quad - 0.47(0.73\log_2(0.73)) \\ &\approx 0.57 \end{aligned}$$

Similarly, $H(\boldsymbol{Y} \mid \boldsymbol{X}) \approx 0.94$.

B.2 Independent Random Variables

In the previous section, the random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ are not independent. For example, $p_{\boldsymbol{XY}}(0,0) = 1/32 = 32/1024$ which is different from $p_{\boldsymbol{X}}(0) \times p_{\boldsymbol{Y}}(0) = 5/32 \times 17/32 = 85/1024$. Intuitively, it may seem plausible that the knowledge of a linear combination between two variables, not independent, reveals information. It is shown in this section, that the knowledge of a linear combination between two independent variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ may still reveal some information.

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two discrete random variables with alphabet $\mathbb{K} = \mathrm{GF}(3)$ and let $p_{\boldsymbol{XY}}$ be their joint probability mass function, defined in Table B.1.

Table B.1: Joint Probability Mass Function $p_{\boldsymbol{XY}}$

$\boldsymbol{Y}$	0	1	2
$\boldsymbol{X}$			
0	5/100	10/100	10/100
1	5/100	10/100	10/100
2	10/100	20/100	20/100

The marginal probability mass functions $p_{\boldsymbol{X}}$ and $p_{\boldsymbol{Y}}$ are defined by:

$$\begin{cases} p_{\boldsymbol{X}}(0) = p_{\boldsymbol{X}}(1) = 5/100 + 10/100 + 10/100 = 25/100 \\ p_{\boldsymbol{X}}(2) = 10/100 + 20/100 + 20/100 = 50/100 \\ p_{\boldsymbol{Y}}(0) = 5/100 + 5/100 + 10/100 = 20/100 \\ p_{\boldsymbol{Y}}(1) = p_{\boldsymbol{Y}}(2) = 10/100 + 10/100 + 20/100 = 40/100 \end{cases}$$

The corresponding entropies are then:

$$\begin{cases} H(\boldsymbol{X}) = -p_{\boldsymbol{X}}(0)\log_2(p_{\boldsymbol{X}}(0)) \\ \qquad\quad - p_{\boldsymbol{X}}(1)\log_2(p_{\boldsymbol{X}}(1)) \\ \qquad\quad - p_{\boldsymbol{X}}(2)\log_2(p_{\boldsymbol{X}}(2)) \approx 1.50 & \text{(B.5)} \\ H(\boldsymbol{Y}) = -p_{\boldsymbol{Y}}(0)\log_2(p_{\boldsymbol{Y}}(0)) \\ \qquad\quad - p_{\boldsymbol{Y}}(1)\log_2(p_{\boldsymbol{Y}}(1)) \\ \qquad\quad - p_{\boldsymbol{Y}}(2)\log_2(p_{\boldsymbol{Y}}(2)) \approx 1.52 & \text{(B.6)} \end{cases}$$

It is assumed that a party obtains the pair $[\, a, b \,] \in \mathbb{K}^2$ as well as the linear combination $z = ax + by$ (again, the random variable associated with the linear combination z is denoted by $\boldsymbol{Z}$). For example, $a = 1$, $b = 1$ and $z = 0$. Since $z = 0$, either $[\, x, y \,] = [\, 0, 0 \,]$,

$[\,x, y\,] = [\,1, 2\,]$ or $[\,x, y\,] = [\,2, 1\,]$. It follows:

$$\begin{cases} p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0) \\ \quad = \dfrac{p_{\boldsymbol{XY}}(0,0)}{p_{\boldsymbol{XY}}(0,0) + p_{\boldsymbol{XY}}(1,2) + p_{\boldsymbol{XY}}(2,1)} \\ \quad = \dfrac{5}{5+10+20} = 1/7 \\ p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0) \\ \quad = \dfrac{p_{\boldsymbol{XY}}(1,2)}{p_{\boldsymbol{XY}}(0,0) + p_{\boldsymbol{XY}}(1,2) + p_{\boldsymbol{XY}}(2,1)} \\ \quad = \dfrac{10}{5+10+20} = 2/7 \\ p_{\boldsymbol{X}|\boldsymbol{Z}}(2 \mid 0) \\ \quad = \dfrac{p_{\boldsymbol{XY}}(2,1)}{p_{\boldsymbol{XY}}(0,0) + p_{\boldsymbol{XY}}(1,2) + p_{\boldsymbol{XY}}(2,1)} \\ \quad = \dfrac{20}{5+10+20} = 4/7 \end{cases}$$

The conditional entropy $H(\boldsymbol{X} \mid \boldsymbol{Z})$ is

$$\begin{aligned} H(\boldsymbol{X} \mid \boldsymbol{Z}) &= p_{\boldsymbol{Z}}(0)H(\boldsymbol{X} \mid \boldsymbol{Z} = 0) + p_{\boldsymbol{Z}}(1)H(\boldsymbol{X} \mid \boldsymbol{Z} = 1) + p_{\boldsymbol{Z}}(2)H(\boldsymbol{X} \mid \boldsymbol{Z} = 2) \\ &= H(\boldsymbol{X} \mid \boldsymbol{Z} = 0) \\ &= -p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0)\log_2(p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0)) \\ &\quad - p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0)\log_2(p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0)) \\ &\quad - p_{\boldsymbol{X}|\boldsymbol{Z}}(2 \mid 0)\log_2(p_{\boldsymbol{X}|\boldsymbol{Z}}(2 \mid 0)) \\ &\approx 1.38 \end{aligned} \tag{B.7}$$

Also, using the equalities

$$\begin{cases} p_{\boldsymbol{X}|\boldsymbol{Z}}(0 \mid 0) = p_{\boldsymbol{Y}|\boldsymbol{Z}}(0 \mid 0) \\ p_{\boldsymbol{X}|\boldsymbol{Z}}(1 \mid 0) = p_{\boldsymbol{Y}|\boldsymbol{Z}}(2 \mid 0) \text{ and} \\ p_{\boldsymbol{X}|\boldsymbol{Z}}(2 \mid 0) = p_{\boldsymbol{Y}|\boldsymbol{Z}}(1 \mid 0), \end{cases}$$

one may observe, from (B.7), that

$$H(\boldsymbol{Y} \mid \boldsymbol{Z}) = H(\boldsymbol{X} \mid \boldsymbol{Z}) \approx 1.38. \tag{B.8}$$

From (B.5) and (B.7), it is concluded that $H(\boldsymbol{X} \mid \boldsymbol{Z}) < H(\boldsymbol{X})$ and from (B.6) and (B.8), the inequality $H(\boldsymbol{Y} \mid \boldsymbol{Z}) < H(\boldsymbol{Y})$is obtained. This demonstrates that the knowledge of a linear combination between $\boldsymbol{X}$ and $\boldsymbol{Y}$ leaks information.

Again, note that neither $H(\boldsymbol{X} \mid \boldsymbol{Y}) = 0$ nor $H(\boldsymbol{Y} \mid \boldsymbol{X}) = 0$. Indeed, because $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent, $H(\boldsymbol{X} \mid \boldsymbol{Y}) = H(\boldsymbol{X}) \approx 1.50$ and $H(\boldsymbol{Y} \mid \boldsymbol{X}) = H(\boldsymbol{Y}) \approx 1.52$.

Appendix C

Characteristics of Some Polynomial Interpolation-based Distributed Oblivious Transfer Protocols

This appendix presents the technical characteristics of the DOT protocols described in Sect. 1.3. Security parameters are defined in Sect. 2.2.3 while protocols' parameters are detailed in Sect. 3.3.

C.1 Naor and Pinkas' Distributed Oblivious Transfer Protocol

In the sparse polynomial interpolation-based DOT protocol presented in 2000 by Naor and Pinkas [74], the number of secrets is $n = 2$, the threshold parameter is k, the sender's privacy and strong privacy parameters are respectively $\lambda_S = k - 1$ 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.

C.2 Blundo et al.'s Distributed Oblivious Transfer Protocols

C.2.1 Original Distributed Oblivious Transfer Protocol

The protocol introduced by Blundo, D'Arco, De Santis, and Stinson [19] is an extension of Naor and Pinkas' sparse polynomial interpolation-based DOT protocol [74]. Only the following characteristics are different from those described in Sect. C.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 $i = 1, 2, \ldots, N$.

C.2.2 Improved Distributed Oblivious Transfer Protocol

The characteristics of the protocol designed by Blundo, D'Arco, De Santis, and Stinson [20] in 2007 are similar to those of the protocol they presented in 2002 (see Sect. C.2.1). 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, using Shamir's secret sharing scheme [89],
- 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 Sect. C.2.1.

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 $F_t^{(\ell)}$ is the sharing polynomial determined in Shamir's secret sharing scheme to share $r_t^{(\ell)}$, the share $F_t^{(\ell)}(j)$ allocated to server S_j is denoted by $[r_t^{(\ell)}]_j$.)

In the transfer phase, on reception of the request z_j from the receiver $\mathcal{R}$, a server S_j $(j \in \mathcal{I})$ calculates $\boldsymbol{v}_j = \left[V_1(z_j), V_2(z_j), \ldots, V_N(z_j) \right]$ and returns $\boldsymbol{u}_j^{(1)} \cdot \boldsymbol{v}_j$ and $\boldsymbol{u}_j^{(2)} \cdot \boldsymbol{v}_j$, with the two sets of $n-1$ shares, $\left[[r_2^{(1)}]_j, [r_3^{(1)}]_j, \ldots, [r_n^{(1)}]_j \right]$ and $\left[[r_2^{(2)}]_j, [r_3^{(2)}]_j, \ldots, [r_n^{(2)}]_j \right]$, to $\mathcal{R}$. From the collected values $\boldsymbol{u}_j^{(1)} \cdot \boldsymbol{v}_j$, $\mathcal{R}$ interpolates a first polynomial $R^{(1)}$. Similarly, from the collected values $\boldsymbol{u}_j^{(2)} \cdot \boldsymbol{v}_j$, $\mathcal{R}$ interpolates a second polynomial $R^{(2)}$. Then, if the choice of $\mathcal{R}$ is $\sigma = 1$, then $\mathcal{R}$ 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.

C.3 Nikov et al.'s Distributed Oblivious Transfer Protocol

The sparse polynomial interpolation-based DOT protocol introduced by Nikov, Nikova, Preneel, and Vandewalle [78] is characterized by the following parameters: the number of secrets $n > 1$, 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 strong privacy 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 $i = 1, 2, \ldots, N$.

As in Naor and Pinkas' and in Blundo et al.'s DOT protocols (see Sect. C.1 and C.2.1 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.

C.4 Beimel et al.'s Distributed Oblivious Transfer Protocol

In [8], 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 DOT protocol are: the number of secrets $n > 1$, 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:

- $U_1(x) = \sum_{i=0}^{i=k-1} a_{1,i} x^i$, 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 $\ell \in [\![1, n]\!]$.

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 $\boldsymbol{u}_j \cdot \boldsymbol{v}_j$ 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.

Appendix D

Jiang, Li, and Li's Distributed Oblivious Transfer Protocol Analysis

Jiang, Li, and Li have proposed an unconditionally secure adaptive DOT protocol [65] based on Nikov, Nikova, Preneel, and Vandewalle's DOT protocol [78].

Let $\mathbb{K} = \mathrm{GF}(p)$ ($p \in \mathbb{N}$, p prime) be a finite field. Nikov et al.'s DOT protocol encompasses a sender $\mathcal{S}$ who owns n secrets $\omega_0, \omega_1, \ldots, \omega_{n-1}$ selected in $\mathbb{K}$ ($n > 1$), a receiver $\mathcal{R}$ who wishes to learn a secret ω_σ ($\sigma \in [\![0, n-1]\!]$), and m servers S_i ($i \in \mathcal{I}_m \subseteq \mathbb{K}^*$, $|\mathcal{I}_m| = m$, $m > 1$) by contacting k servers ($1 < k \leq m$). It is assumed that $p > \max(n, m)$ and that all operations are executed in $\mathbb{K}$. In addition, assuming that $\lambda_\mathrm{R} + \lambda_\mathrm{C} \leq k - 1$ and $\lambda_\mathrm{S} \leq \lambda_\mathrm{C}$, the protocol satisfies security conditions C_1, C_2 with $\lambda_\mathrm{R} \leq k - 1$, C_3 with $\lambda_\mathrm{S} \leq k - 1$ and C_4 with $\lambda_\mathrm{C} \leq k - 1$ (security conditions and parameters λ_R, λ_S and λ_C are described in Sect. 2.2.3).

Actually, Nikov et al.'s DOT protocol is similar to Blundo, D'Arco, De Santis, and Stinson's DOT protocol [19], described in Sect. 2.3.2 (see the protocol's characteristics in App. C.3). The main difference between both protocols lies in the form of the polynomial generated in the set-up phase; in Nikov et al.'s protocol, the sender $\mathcal{S}$ generates a quasi-random polynomial B_0 of $\mathbb{K}_{k-1}[X]$, such that $B_0(0) = \omega_0$ and $n - 1$ quasi-random polynomials B_i ($1 \leq i \leq n - 1$), of $\mathbb{K}_{\lambda_\mathrm{C}}[X]$ ($0 \leq \lambda_\mathrm{C} < k$), such that $B_i(0) = \omega_i - \omega_0$. Then, $\mathcal{S}$ generates a sparse n-variate polynomial Q defined by

$$Q(x, y_1, y_2, \ldots, y_{n-1}) = B_0(x) + \sum_{j=1}^{n-1} B_j(x) \times y_j \,.$$

In the protocol proposed by Jiang et al., the set-up phase is unchanged and the transfer phase, where the polynomials Z_i $(1 \leq i \leq n-1)$ belong to $\mathbb{K}_{\lambda_R}[X]$, is repeated t times, allowing the receiver $\mathcal{R}$ to obtain t secrets.

Jiang et al. describe a 'special case' allowing $\mathcal{R}$ to determine any polynomial B_j $(1 \leq j \leq n-1)$ and claim that it does not reduce the sender's security level. However, what is important is the number of contacted servers required to determine B_j: only $\lambda_C + 1$ instead of k, which leaves $\mathcal{R}$ with $k - (\lambda_C + 1)$ redundant shares that can be used to determine an additional secret. Thus, assuming the set-up phase is the same as in Sects. 7.4 and 7.5, and that the ℓ-th adaptive transfer phase is denoted by '*run* $\ell - 1$', it is possible to devise the following attack:

In run 0, the dishonest receiver $\mathcal{R}$ selects the secret index $\sigma = 0$ and transmits to each server S_i $(i \in \mathcal{I}_k = \{ i_1, i_2, \ldots, i_k \})$ the request $\boldsymbol{q} = [0, 0, \ldots, 0]$ composed of $n-1$ zeros. The k collected shares are then $B_0(i_1), B_0(i_2), \ldots, B_0(i_k)$. Since $B_0 \in \mathbb{K}_{k-1}[X]$, $\mathcal{R}$ is able to calculate B_0 and also $\omega_\sigma = \omega_0 = B_0(0)$, thanks to Lagrange's interpolation technique (see Sect. 2.1.1.3).

It is assumed that $\mathcal{R}$ wishes to obtain the secret ω_{σ_ℓ} in run ℓ and in addition, the secret ω_{σ_+}. Each run ℓ, from run 1, allows $\mathcal{R}$ to obtain $\lambda_C + 1$ shares to calculate the chosen secret ω_{σ_ℓ}, as well as $u = k - (\lambda_C + 1)$ extra shares. It follows that the number of runs to obtain ω_{σ_+} is $v = 1 + \lceil \frac{\lambda_C+1}{u} \rceil = \lceil \frac{k}{k-(\lambda_C+1)} \rceil$.

For each run ℓ $(1 \leq \ell \leq v-1)$, $\mathcal{R}$ prepares two lists of servers: $\mathcal{J}^{(\ell)}$, the list of $\lambda_C + 1$ servers to contact to obtain the secret ω_{σ_ℓ} and $\mathcal{J}_+^{(\ell)}$ the list of $k - (\lambda_C + 1)$ servers to contact to obtain shares of ω_{σ_+}. The list $\mathcal{J}_+^{(\ell)}$ may be defined for example by $\mathcal{J}_+^{(\ell)} = \{ i_{(\ell-1)\times(k-\lambda_C+1)+1}, i_{(\ell-1)\times(k-\lambda_C+1)+2}, \ldots, i_{\ell\times(k-\lambda_C+1)} \}$. The set $\mathcal{J}^{(\ell)}$ is composed of the rest of the list of servers, i.e., $\mathcal{J}^{(\ell)} = \mathcal{I}_k \setminus \mathcal{J}_+^{\ell}$.

Each server of $\mathcal{J}^{(\ell)}$ receives a request $\boldsymbol{q}^{\ell} = \left[\delta_1^{\sigma_\ell}, \delta_2^{\sigma_\ell}, \ldots, \delta_{n-1}^{\sigma_\ell} \right]$ whereas each server of $\mathcal{J}_+^{(\ell)}$ receives a request $\boldsymbol{q}^{+} = \left[\delta_1^{\sigma_+}, \delta_2^{\sigma_+}, \ldots, \delta_{n-1}^{\sigma_+} \right]$.

A server $S_j \in \mathcal{J}^{(\ell)}$ responds with $Q(j, \boldsymbol{q}^{\ell}) = B_0(j) + B_{\sigma_\ell}(j)$. Because $\mathcal{R}$ determined B_0 in run 0, she is able to calculate $B_{\sigma_\ell}(j)$. The polynomial B_{σ_ℓ} belongs to $\mathbb{K}_{\lambda_C}[X]$; therefore, using Lagrange's interpolation technique, $\mathcal{R}$ is able to determine B_{σ_ℓ} from the $\lambda_C + 1$ shares collected from the servers of $\mathcal{J}^{(\ell)}$ and hence to determine $\omega_{\sigma_\ell} = B_0(0) + B_{\sigma_\ell}(0)$.

At the end of run $v-1$, $\mathcal{R}$ has collected $\lambda_C + 1$ shares $Q(j, \boldsymbol{q}^{+}) = B_0(j) + B_{\sigma_+}(j)$ from different servers S_j of $\mathcal{J}_+^{(1)} \cup \mathcal{J}_+^{(2)} \cup \cdots \cup \mathcal{J}_+^{(v-1)}$. The polynomial B_0 being known, $\mathcal{R}$ can determine the $\lambda_C + 1$ values $B_{\sigma_+}(j)$. Since the polynomial B_{σ_+} belongs to $\mathbb{K}_{\lambda_C}[X]$, $\mathcal{R}$ is able to apply again Lagrange's interpolation technique and determine $B_{\sigma_+}(0)$. The last step consists in calculating $\omega_{\sigma_+} = B_0(0) + B_{\sigma_+}(0)$.

In conclusion, if the receiver does not follow the protocol, she is able to obtain one secret per run plus another secret at the end of run v. This attack shows that, without modifications, Nikov et al.'s DOT protocol cannot be used as an adaptive DOT protocol.

Appendix E

A Few Demonstrations

E.1 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 the following theorem:

Theorem E.1. *The linear systems* (5.1) *and* (5.2) *determined in Sect. 5.7.3.1 are identical.*

Proof.
System (5.2) can be written under the matrix form $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$, where

$$\mathcal{A} = \begin{pmatrix} x_1 & x_1^2 & \dots & x_1^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_1)) & Z_1(x_1) & Z_2(x_1) & \dots & Z_{n-1}(x_1) \\ x_2 & x_2^2 & \dots & x_2^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_2)) & Z_1(x_2) & Z_2(x_2) & \dots & Z_{n-1}(x_2) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ x_k & x_k^2 & \dots & x_k^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_k)) & Z_1(x_k) & Z_2(x_k) & \dots & Z_{n-1}(x_k) \\ x_{k+1} & x_{k+1}^2 & \dots & x_{k+1}^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_{k+1})) & Z_1(x_{k+1}) & Z_2(x_{k+1}) & \dots & Z_{n-1}(x_{k+1}) \\ x_{k+2} & x_{k+2}^2 & \dots & x_{k+2}^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_{k+2})) & Z_1(x_{k+2}) & Z_2(x_{k+2}) & \dots & Z_{n-1}(x_{k+2}) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ x_c & x_c^2 & \dots & x_c^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_c)) & Z_1(x_c) & Z_2(x_c) & \dots & Z_{n-1}(x_c) \end{pmatrix},$$

$$\boldsymbol{x} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{k-1} \\ \omega_0 \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{pmatrix}, \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} \mu_{x_1} \\ \mu_{x_2} \\ \vdots \\ \mu_{x_k} \\ \mu_{x_{k+1}} \\ \mu_{x_{k+2}} \\ \vdots \\ \mu_{x_c} \end{pmatrix}.$$

We show that each row ℓ ($k+1 \leq \ell \leq c$) of $\mathcal{A}$ 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 $i = 1, 2, \ldots, k-1$,
- $T_k = 1 - \sum_{j=1}^{n-1} Z_j$,
- $T_{k+s} = Z_s$ for $s = 1, 2, \ldots, n-1$.

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

$$\mathcal{A} = \begin{pmatrix} T_1(x_1) & T_2(x_1) & \ldots & T_{k-1}(x_1) & T_k(x_1) & T_{k+1}(x_1) & T_{k+2}(x_1) & \ldots & T_{k+n-1}(x_1) \\ T_1(x_2) & T_2(x_2) & \ldots & T_{k-1}(x_2) & T_k(x_2) & T_{k+1}(x_2) & T_{k+2}(x_2) & \ldots & T_{k+n-1}(x_2) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ T_1(x_c) & T_2(x_c) & \ldots & T_{k-1}(x_c) & T_k(x_c) & T_{k+1}(x_c) & T_{k+2}(x_c) & \ldots & T_{k+n-1}(x_c) \end{pmatrix},$$

In $\mathbb{K}_{k-1}[X]$, we consider the k Lagrange polynomials L_{x_i} related to $x_1, x_2, \ldots, x_k$ (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 $\left[L_{x_1}, L_{x_2}, \ldots, L_{x_k} \right]$ 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 $\mathcal{A}$ is:

$$\mathcal{A} = \begin{pmatrix} T_1(x_1) & T_2(x_1) & \dots & T_{k+n-1}(x_1) \\ T_1(x_2) & T_2(x_2) & \dots & T_{k+n-1}(x_2) \\ \vdots & \vdots & & \vdots \\ T_1(x_k) & T_2(x_k) & \dots & T_{k+n-1}(x_k) \\ \sum_{j=1}^{k} T_1(x_j)L_{x_j}(x_{k+1}) & \sum_{j=1}^{k} T_2(x_j)L_{x_j}(x_{k+1}) & \dots & \sum_{j=1}^{k} T_{k+n-1}(x_j)L_{x_j}(x_{k+1}) \\ \sum_{j=1}^{k} T_1(x_j)L_{x_j}(x_{k+2}) & \sum_{j=1}^{k} T_2(x_j)L_{x_j}(x_{k+2}) & \dots & \sum_{j=1}^{k} T_{k+n-1}(x_j)L_{x_j}(x_{k+2}) \\ \vdots & \vdots & & \vdots \\ \sum_{j=1}^{k} T_1(x_j)L_{x_j}(x_c) & \sum_{j=1}^{k} T_2(x_j)L_{x_j}(x_c) & \dots & \sum_{j=1}^{k} T_{k+n-1}(x_j)L_{x_j}(x_c) \end{pmatrix}.$$

Clearly, the lines $k+1, k+2, \dots, c$ are linear combinations of the first k lines.
It follows that the matrix equality $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ can be written $\mathcal{A}'\boldsymbol{x} = \boldsymbol{b}'$, where

$$\mathcal{A}' = \begin{pmatrix} x_1 & x_1^2 & \dots & x_1^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_1)) & Z_1(x_1) & Z_2(x_1) & \dots & Z_{n-1}(x_1) \\ x_2 & x_2^2 & \dots & x_2^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_2)) & Z_1(x_2) & Z_2(x_2) & \dots & Z_{n-1}(x_2) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ x_k & x_k^2 & \dots & x_k^{k-1} & (1 - \sum_{j=1}^{n-1} Z_j(x_k)) & Z_1(x_k) & Z_2(x_k) & \dots & Z_{n-1}(x_k) \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

and

$$\boldsymbol{b}' = \begin{pmatrix} \mu_{x_1} \\ \mu_{x_2} \\ \vdots \\ \mu_{x_k} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The linear system corresponding to this equation is exactly (5.1), which concludes the demonstration. □

E.2 Conditional Entropy with Fixed Condition

The objective of this section is to prove that if $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ are three random variables over respective alphabets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ such that $\boldsymbol{X}$ and $[\,\boldsymbol{Y},\boldsymbol{Z}\,]$ are independent, then $H(\boldsymbol{X} \mid \boldsymbol{Y},\boldsymbol{Z}) = H(\boldsymbol{X}) = H(\boldsymbol{X} \mid \boldsymbol{Y},\boldsymbol{Z} = z)$ for $z \in \mathcal{Z}$.

Before a demonstration is given, a few properties on probabilities and discrete random variables are introduced.

Lemma E.2. *Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three random variables over respective alphabets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$. Their joint probability mass function is denoted by $p_{\boldsymbol{XYZ}}$. If $[\,x,y,z\,] \in \mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$ with $p_{\boldsymbol{Z}}(z) \neq 0$,*

$$p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z) = \sum_{x\in\mathcal{X}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x,y \mid z)\,.$$

Proof.

$$\begin{aligned}
p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z) &= p_{\boldsymbol{YZ}}(y,z)/p_{\boldsymbol{Z}}(z) && \text{(from (2.2))}\\
&= \left(\sum_{x\in\mathcal{X}} p_{\boldsymbol{XYZ}}(x,y,z)\right)/p_{\boldsymbol{Z}}(z) && \text{(from (2.1))}\\
&= \sum_{x\in\mathcal{X}} p_{\boldsymbol{XYZ}}(x,y,z)/p_{\boldsymbol{Z}}(z)\\
&= \sum_{x\in\mathcal{X}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x,y \mid z) && \text{(from (2.2))}
\end{aligned}$$

□

Lemma E.3. *Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three random variables over respective alphabets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$. Their joint probability mass function is denoted by $p_{\boldsymbol{XYZ}}$. If $z \in \mathcal{Z}$,*

$$H(\boldsymbol{X} \mid \boldsymbol{Y},\boldsymbol{Z} = z) = -\sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x,y \mid z)\log_2(p_{\boldsymbol{X}|\boldsymbol{YZ}}(x \mid y,z))\,.$$

Proof.
Using (2.4), we develop $H(\boldsymbol{X} \mid \boldsymbol{Y},\boldsymbol{Z} = z)$:

$$H(\boldsymbol{X} \mid \boldsymbol{Y},\boldsymbol{Z} = z) = H(\boldsymbol{X},\boldsymbol{Y} \mid \boldsymbol{Z} = z) - H(\boldsymbol{Y} \mid \boldsymbol{Z} = z)\,.$$

On one hand, from the definition of the conditional entropy, we can write

$$H(\boldsymbol{X},\boldsymbol{Y} \mid \boldsymbol{Z} = z) = -\sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x,y \mid z)\log_2(p_{\boldsymbol{XY}|\boldsymbol{Z}}(x,y \mid z))\,.$$

On the other hand:

$$\begin{aligned} H(\boldsymbol{Y} \mid \boldsymbol{Z} = z) & \\ &= -\sum_{y \in \mathcal{Y}} p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z) \log_2(p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z)) \\ &= -\sum_{y \in \mathcal{Y}} \left(\sum_{x \in \mathcal{X}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z) \right) \log_2(p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z)) \qquad \text{(from Lemma E.2)} \\ &= -\sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z) \log_2(p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z)) \end{aligned}$$

Note that if $p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z) = 0$, then by convention $p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z) \log_2(p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z)) = 0$. Therefore, we can assume that the pairs $[\, y, z \,] \in \mathcal{Y} \times \mathcal{Z}$ are such that $p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z) \neq 0$. It follows

$$\begin{aligned} H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z} = z) & \\ &= -\sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z) \left(\log_2(p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z)) - \log_2(p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z)) \right) \\ &= -\sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z) \log_2(p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z) / p_{\boldsymbol{Y}|\boldsymbol{Z}}(y \mid z)) \\ &= -\sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z) \log_2(p_{\boldsymbol{X}|\boldsymbol{YZ}}(x \mid y, z)) \qquad \text{(from (2.2))} \end{aligned}$$

□

Formally, it is possible to demonstrate now the following theorem:

Theorem E.4. *Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three random variables over respective alphabets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$. If $H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = H(\boldsymbol{X})$ then for $z_i \in \mathcal{Z}$ such that $p_{\boldsymbol{Z}}(z_i) \neq 0$, the equality $H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z} = z_i) = H(\boldsymbol{X})$ holds.*

Proof.
We denote $p_{\boldsymbol{XYZ}}$ the joint probability mass function associated with the random variables $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$. The probability mass functions $p_{\boldsymbol{X}}, p_{\boldsymbol{Z}}$ and the joint probability mass function $p_{\boldsymbol{YZ}}$ are determined from $p_{\boldsymbol{XYZ}}$ thanks to the relation (2.1). Because $H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z}) = H(\boldsymbol{X})$, the variables $\boldsymbol{X}$ and $[\, \boldsymbol{Y}, \boldsymbol{Z} \,]$ are independent. Their corresponding probability mass functions satisfy the relation

$$p_{\boldsymbol{XYZ}}(x, y, z) = p_{\boldsymbol{X}}(x) p_{\boldsymbol{YZ}}(y, z) \tag{E.1}$$

for $[\, x, y, z \,] \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. Therefore, assuming that $p_{\boldsymbol{YZ}}(y, z) \neq 0$, we have

$$p_{\boldsymbol{X}|\boldsymbol{YZ}}(x \mid y, z) = p_{\boldsymbol{X}}(x)\,. \tag{E.2}$$

Indeed,

$$\begin{aligned} p_{\boldsymbol{X}|\boldsymbol{YZ}}(x \mid y, z) &= p_{\boldsymbol{XYZ}}(x, y, z)/p_{\boldsymbol{YZ}}(y, z) && \text{(from (2.1))} \\ &= p_{\boldsymbol{X}}(x)p_{\boldsymbol{YZ}}(y, z)/p_{\boldsymbol{YZ}}(y, z) && \text{(from (E.1))} \\ &= p_{\boldsymbol{X}}(x) \end{aligned}$$

$$\begin{aligned} &H(\boldsymbol{X} \mid \boldsymbol{Y}, \boldsymbol{Z} = z_i) \\ &\quad = -\sum_{\substack{x\in\mathcal{X} \\ y\in\mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z_i) \log_2(p_{\boldsymbol{X}|\boldsymbol{YZ}}(x \mid y, z_i)) && \text{(from Lemma E.3)} \\ &\quad = -\sum_{\substack{x\in\mathcal{X} \\ y\in\mathcal{Y}}} p_{\boldsymbol{XY}|\boldsymbol{Z}}(x, y \mid z_i) \log_2(p_{\boldsymbol{X}}(x)) && \text{(from (E.2))} \\ &\quad = -\sum_{\substack{x\in\mathcal{X} \\ y\in\mathcal{Y}}} p_{\boldsymbol{XYZ}}(x, y, z_i)/p_{\boldsymbol{Z}}(z_i) \times \log_2(p_{\boldsymbol{X}}(x)) && \text{(from (2.2))} \\ &\quad = -\sum_{\substack{x\in\mathcal{X} \\ y\in\mathcal{Y}}} p_{\boldsymbol{X}}(x)p_{\boldsymbol{YZ}}(y, z_i)/p_{\boldsymbol{Z}}(z_i) \times \log_2(p_{\boldsymbol{X}}(x)) && \text{(from (E.1))} \\ &\quad = \sum_{y\in\mathcal{Y}} p_{\boldsymbol{YZ}}(y, z_i)/p_{\boldsymbol{Z}}(z_i) \times \left(-\sum_{x\in\mathcal{X}} p_{\boldsymbol{X}}(x) \log_2(p_{\boldsymbol{X}}(x)) \right) \\ &\quad = \sum_{y\in\mathcal{Y}} p_{\boldsymbol{YZ}}(y, z_i)/p_{\boldsymbol{Z}}(z_i) \times H(\boldsymbol{X}) \\ &\quad = H(\boldsymbol{X})/p_{\boldsymbol{Z}}(z_i) \times \sum_{y\in\mathcal{Y}} p_{\boldsymbol{YZ}}(y, z_i) \\ &\quad = H(\boldsymbol{X})/p_{\boldsymbol{Z}}(z_i) \times p_{\boldsymbol{Z}}(z_i) && \text{(from (2.1))} \\ &\quad = H(\boldsymbol{X}) \end{aligned}$$

□

E.3 Maximal Matching in a Graph

A *graph* $\mathcal{G}$ is defined as a pair $[\,V, E\,]$ where $V = \{\,v_1, v_2, \ldots, v_n\,\}$ ($n \in \mathbb{N}$) is a set of *vertices* or nodes and $E = \{\,e_1, e_2, \ldots, e_m\,\}$ ($m \in \mathbb{N}$) is a set of *edges*, where an edge is a pair of vertices $\left[\,v_i, v_j\,\right]$ ($v_i \in V$, $v_j \in V$, $v_i \neq v_j$).

The graph $\bar{\mathcal{G}} = \left[\,\bar{V}, \bar{E}\,\right]$ 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* $M \subseteq E$ 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: 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.

Appendix F

Example of Insecurity in Blundo et al.'s Distributed Oblivious Transfer Protocol

This appendix presents an example proving that the polynomial interpolation-based DOT protocol introduced by Blundo, D'Arco, De Santis, and Stinson [20] is insecure. More precisely, in the scenario described below, a receiver is able to determine two secrets although she is entitled to obtain one secret only.

The example can be decomposed into three sequential parts. First, the values of the public information shared by the sender, the receiver and the servers involved in the protocol are given. Second, the information generated in the set-up phase is described: the information private to the sender and the information received by the servers. Third, the values of the information generated in the transfer phase are presented: the choice of the receiver and the requests received by the servers. Lastly, it is shown how the receiver, once she has collected the responses from the contacted servers, can determine more than one secret. The protocol is described in Sect. 2.3.3 and its characteristics are described in App. C.2.2.

F.1 Public Information

The parameters of the protocol are:

- Finite field $\mathbb{K} = \mathrm{GF}(11)$
- Threshold $k = 3$
- Number of secrets $n = 2$ (ω_1 and ω_2)
- Number of servers $m = 3$ (S_1, S_2, and S_3)

- $\mathbf{\Omega_1}$ (resp. $\mathbf{\Omega_2}$) is the random variable associated with ω_1 (resp. ω_2)

The probability mass functions $p_{\mathbf{\Omega_1}}$ and $p_{\mathbf{\Omega_2}}$, associated with the random variables $\mathbf{\Omega_1}$ and $\mathbf{\Omega_2}$, are defined below.

- $p_{\mathbf{\Omega_1}}(6) = 0.5$, $p_{\mathbf{\Omega_1}}(2) = 0.5$, $p_{\mathbf{\Omega_1}}(i) = 0$ if $i \neq 6$ and $i \neq 2$
- $p_{\mathbf{\Omega_2}}(1) = 0.5$, $p_{\mathbf{\Omega_2}}(3) = 0.5$, $p_{\mathbf{\Omega_2}}(i) = 0$ if $i \neq 1$ and $i \neq 3$

F.2 Set-up Phase

F.2.1 Information Private to the Sender

The secrets held by the sender are $\omega_1 = 6$ and $\omega_2 = 1$. The sender also randomly selects the following masks:

- $c_1 = 5$ and $c_2 = 7$,
- $r_1^{(1)} = 1$ and $r_2^{(1)} = 2$, and
- $r_1^{(2)} = 4$ and $r_2^{(2)} = 5$.

The masks c_1 and c_2 are selected in $\mathbb{K}^*$ while the masks $r_1^{(1)}$, $r_2^{(1)}$, $r_1^{(2)}$, and $r_2^{(2)}$ are selected in $\mathbb{K}$.

F.2.2 Intermediate Computation to Prepare the Sharing Polynomials

The sender prepares two sharing polynomials $U_1^{(1)}$ and $U_1^{(2)}$, and distributes the corresponding shares to servers S_1, S_2, and S_3. The polynomials, shares and intermediate computation results are detailed below.

- $r_1^{(1)} \times c_1 \times \omega_1 = 8$,
- $r_2^{(1)} \times c_2 \times \omega_2 = 3$,
- $r_1^{(2)} \times c_1 = 9$,
- $r_2^{(2)} \times c_2 = 2$,
- 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 $\boldsymbol{u}_2^{(2)} = [\,5, 4\,]$, and
- S_3 receives $\boldsymbol{u}_3^{(1)} = [\,7, 6\,]$ and $\boldsymbol{u}_3^{(2)} = [\,4, 4\,]$.

F.3 Transfer Phase

In the transfer phase, the receiver generates a request and sends the corresponding shares to the servers S_1, S_2, and S_3. Each server elaborates a response which is returned to the receiver.

- Request: $z_j = [\,1, 6\,]$ $(j = 1, 2, 3)$,
- $\boldsymbol{v}_j = \boldsymbol{z}_j = [\,1, 6\,]$,
- S_1 replies with $\boldsymbol{u}_1^{(1)} \cdot \boldsymbol{v}_1 = 0$ and $\boldsymbol{u}_1^{(2)} \cdot \boldsymbol{v}_1 = 5$,
- S_2 replies with $\boldsymbol{u}_2^{(1)} \cdot \boldsymbol{v}_2 = 7$ and $\boldsymbol{u}_2^{(2)} \cdot \boldsymbol{v}_2 = 7$, and
- S_3 replies with $\boldsymbol{u}_3^{(1)} \cdot \boldsymbol{v}_3 = 10$ and $\boldsymbol{u}_3^{(2)} \cdot \boldsymbol{v}_3 = 6$.

F.4 Obtaining Secrets

The receiver tries to obtain all secrets:

- Interpolated polynomial: $R^{(1)} = 2X + 9X^2$
- $r_2^{(1)} c_2 \omega_2 + r_1^{(1)} c_1 \omega_1 = 2 \times R^{(1)}(0) = 0$
- Interpolated polynomial: $R^{(2)} = X + 4X^2$
- $r_2^{(2)} c_2 + r_1^{(2)} c_1 = 2 \times R^{(2)}(0) = 0$
- 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 system of 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: $c_1 \times (5 \times \omega_2 + 1 \times \omega_1) = 0$, i.e., $5 \times \omega_2 + 1 \times \omega_1 = 0$ since c_1 belongs to $\mathbb{K}^*$ and $\mathbb{K}$ is a field. 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] B. Aiello, Y. Ishai, and O. Reingold. 'Priced Oblivious Transfer: How to Sell Digital Goods'. In: *Advances in Cryptology - EUROCRYPT 2001*. Ed. by B. Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2001, pp. 119–135.

[2] A. Ambainis. 'Upper Bound on Communication Complexity of Private Information Retrieval'. In: *Automata, Languages and Programming - ICALP'97*. Ed. by P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela. Vol. 1256. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1997, pp. 401–407.

[3] O. Barkol, Y. Ishai, and E. Weinreb. 'On Locally Decodable Codes, Self-correctable Codes, and t-Private PIR'. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - APPROX 2007 / RANDOM 2007*. Ed. by M. Charikar, K. Jansen, O. Reingold, and J. D. P. Rolim. Vol. 4627. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2007, pp. 311–325.

[4] O. Barkol, Y. Ishai, and E. Weinreb. 'On Locally Decodable Codes, Self-correctable Codes, and t-Private PIR'. In: *Algorithmica* 58.4 (2010), pp. 831–859.

[5] D. Beaver. 'Commodity-based Cryptography (Extended Abstract)'. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing - STOC'97*. New York, NY, USA: ACM, 1997, pp. 446–455.

[6] D. Beaver. 'Multiparty Protocols Tolerating Half Faulty Processors'. In: *Advances in Cryptology - CRYPTO'89*. Ed. by G. Brassard. Vol. 435. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1990, pp. 560–572.

[7] A. Beimel. 'Secret-Sharing Schemes: A Survey'. In: *Coding and Cryptology - IWCC 2011*. Ed. by Y. M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing. Vol. 6639. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2011, pp. 11–46.

[8] A. Beimel, Y. M. Chee, H. Wang, and L. F. Zhang. 'Communication-efficient Distributed Oblivious Transfer'. In: *Journal of Computer and System Sciences* 78.4 (2012), pp. 1142–1157.

[9] A. Beimel, Y. Ishai, and E. Kushilevitz. 'General constructions for information-theoretic private information retrieval'. In: *Journal of Computer and System Sciences* 71.2 (2005), pp. 213–247.

[10] A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond. 'Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval'. In: *Foundations of Computer Science - FOCS 2002*. IEEE, 2002, pp. 261–270.

[11] M. Bellare and S. Micali. 'Non-interactive Oblivious Transfer and Applications'. In: *Advances in Cryptology - CRYPTO'89*. Ed. by G. Brassard. Vol. 435. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1990, pp. 547–557.

[12] C. H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. 'Practical Quantum Oblivious Transfer'. In: *Advances in Cryptology - CRYPTO'91*. Ed. by J. Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1992, pp. 351–366.

[13] M. Ben-Or, R. Canetti, and O. Goldreich. 'Asynchronous Secure Computation (Extended Abstract)'. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing - STOC'93*. New York, NY, USA: ACM, 1993, pp. 52–61.

[14] M. Ben-Or, S. Goldwasser, and A. Wigderson. 'Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)'. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing - STOC'88*. New York, NY, USA: ACM, 1988, pp. 1–10.

[15] E. R. Berlekamp. *Algebraic Coding Theory*. Revised. Aegean Park Pr, 1984.

[16] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer-Verlag Berlin Heidelberg, 2009.

[17] P. Billingsley. *Probability and Measure*. 3rd. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1995.

[18] G. R. Blakley. 'Safeguarding cryptographic keys'. In: *International Workshop on Managing Requirements Knowledge*. Vol. 48. Montvale, NJ, USA: AFIPS Press, 1979, pp. 313–317.

[19] C. Blundo, P. D'Arco, A. De Santis, and D. R. Stinson. 'New Results on Unconditionally Secure Distributed Oblivious Transfer (Extended Abstract)'. In: *Selected Areas in Cryptography - SAC 2002*. Ed. by K. Nyberg and H. Heys. Vol. 2595. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2003, pp. 291–309.

[20] C. Blundo, P. D'Arco, A. De Santis, and D. R. Stinson. 'On Unconditionally Secure Distributed Oblivious Transfer'. In: *Journal of Cryptology* 20.3 (2007), pp. 323–373.

[21] N. Bourbaki. *Éléments de Mathématique, Algèbre I, Chapitres 1 à 3*. 2nd, Reprint (1970). Springer, 2007.

[22] N. Bourbaki. *Éléments de Mathématique, Algèbre II, Chapitres 4 à 7*. Reprint (1981). Springer, 2007.

[23] G. Brassard and C. Crépeau. '25 Years of Quantum Cryptography'. In: *SIGACT News* 27.3 (1996), pp. 13–24.

[24] G. Brassard, C. Crépeau, and J.-M. Robert. 'All-or-Nothing Disclosure of Secrets'. In: *Advances in Cryptology - CRYPTO'86*. Ed. by A. M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1987, pp. 234–238.

[25] G. Brassard, C. Crépeau, and J.-M. Robert. 'Information Theoretic Reductions Among Disclosure Problems'. In: *Foundations of Computer Science - FOCS 1986*. IEEE, 1986, pp. 168–173.

[26] G. Brassard, C. Crépeau, and M. Sántha. 'Oblivious Transfers and Intersecting Codes'. In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 1769–1780.

[27] D. Chaum, C. Crépeau, and I. B. Damgård. 'Multiparty Unconditionally Secure Protocols (Abstract)'. In: *Advances in Cryptology - CRYPTO'87*. Ed. by C. Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1988, pp. 462–462.

[28] D. Chaum, I. B. Damgård, and J. van de Graaf. 'Multiparty Computations Ensuring Privacy of Each Party's Input and Correctness of the Result'. In: *Advances in Cryptology - CRYPTO'87*. Ed. by C. Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1988, pp. 87–119.

[29] K.-Y. Cheong, T. Koshiba, and S. Nishiyama. 'Strengthening the Security of Distributed Oblivious Transfer'. In: *Information Security and Privacy - ACISP 2009*. Ed. by C. Boyd and J. González Nieto. Vol. 5594. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2009, pp. 377–388.

[30] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. 'Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults'. In: *Foundations of Computer Science - FOCS 1985*. IEEE, 1985, pp. 383–395.

[31] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. 'Private Information Retrieval'. In: *Foundations of Computer Science - FOCS 1995*. IEEE, 1995, pp. 41–50.

[32] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. 'Private Information Retrieval'. In: *Journal of the ACM* 45.6 (1998), pp. 965–981.

[33] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. 2nd ed. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[34] R. Cramer, I. B. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. 'Efficient Multiparty Computations Secure Against an Adaptive Adversary'. In: *Advances in Cryptology - EUROCRYPT'99*. Ed. by J. Stern. Vol. 1592. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1999, pp. 311–326.

[35] R. Cramer, I. B. Damgård, and U. Maurer. 'General Secure Multi-party Computation from any Linear Secret-Sharing Scheme'. In: *Advances in Cryptology - EUROCRYPT 2000*. Ed. by B. Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2000, pp. 316–334.

[36] C. Crépeau. 'Equivalence Between Two Flavours of Oblivious Transfers'. In: *Advances in Cryptology - CRYPTO'87*. Ed. by C. Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1988, pp. 350–354.

[37] C. Crépeau. 'Quantum Oblivious Transfer'. In: *Journal of Modern Optics* 41.12 (1994), pp. 2445–2454.

[38] C. Crépeau. 'Verifiable Disclosure of Secrets and Applications (Abstract)'. In: *Advances in Cryptology - CRYPTO'89*. Ed. by G. Brassard. Vol. 435. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1990, pp. 150–154.

[39] C. Crépeau and J. Kilian. 'Achieving Oblivious Transfer Using Weakened Security Assumptions (Extended Abstract)'. In: *Foundations of Computer Science - FOCS 1988*. IEEE, 1988, pp. 42–52.

[40] P. D'Arco and D. R. Stinson. 'On Unconditionally Secure Robust Distributed Key Distribution Centers'. In: *Advances in Cryptology - ASIACRYPT 2002*. Ed. by Y. Zheng. Vol. 2501. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2002, pp. 346–363.

[41] Y. G. Desmedt and Y. Frankel. 'Shared generation of authenticators and signatures'. In: *Advances in Cryptology - CRYPTO'91*. Ed. by J. Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1992, pp. 457–469.

[42] Y. G. Desmedt and S. Jajodia. *Redistributing secret shares to new access structures and its applications*. Tech. rep. ISSE-TR-97-01. George Mason University, 1997.

[43] Y. G. Desmedt, K. Kurosawa, and T. Van Le. 'Error Correcting and Complexity Aspects of Linear Secret Sharing Schemes'. In: *Information Security - ISC 2003*. Ed. by C. Boyd and W. Wao. Vol. 2851. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2003, pp. 396–407.

[44] B. W. Diffie and M. Hellman. 'New directions in cryptography'. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[45] J. Edmonds. 'Paths, Trees, and Flowers'. In: *Canadian Journal of Mathematics* 17.3 (1965), pp. 449–467.

[46] K. Efremenko. '3-Query Locally Decodable Codes of Subexponential Length'. In: *SIAM Journal on Computing* 41.6 (2012), pp. 1694–1703.

[47] S. Even, O. Goldreich, and A. Lempel. 'A Randomized Protocol for Signing Contracts'. In: *Communications of the ACM* 28.6 (1985), pp. 637–647.

[48] W. Feller. *An Introduction to Probability Theory and Its Applications*. 3rd. Vol. I. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1968.

[49] W. Feller. *An Introduction to Probability Theory and Its Applications*. 2nd. Vol. II. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1971.

[50] M. J. Freedman, K. Nissim, and B. Pinkas. 'Efficient Private Matching and Set Intersection'. In: *Advances in Cryptology - EUROCRYPT 2004*. Ed. by C. Cachin and J. L. Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2004, pp. 1–19.

[51] W. F. Friedman. *The index of coincidence and its applications in cryptography*. Riverbank Publications 22. Geneva, IL, USA: Riverbank Laboratories, 1922.

[52] S. Gao. 'A New Algorithm for Decoding Reed-Solomon Codes'. In: *Communications, Information and Network Security*. Ed. by V. K. Bhargava, H. V. Poor, V. Tarokh, and S. Yoon. Vol. 712. The Springer International Series in Engineering and Computer Science. Springer US, 2003, pp. 55–68.

[53] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. 'The Round Complexity of Verifiable Secret Sharing and Secure Multicast'. In: *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing - STOC'01*. New York, NY, USA: ACM, 2001, pp. 580–589.

[54] Y. Gertner, S. Goldwasser, and T. Malkin. 'A Random Server Model for Private Information Retrieval'. In: *Randomization and Approximation Techniques in Computer Science*. Ed. by M. Luby, J. Rolim, and M. Serna. Vol. 1518. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1998, pp. 200–217.

[55] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. 'Protecting Data Privacy in Private Information Retrieval Schemes'. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing - STOC'98*. New York, NY, USA: ACM, 1998, pp. 151–160.

[56] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. 'Protecting Data Privacy in Private Information Retrieval Schemes'. In: *Journal of Computer and System Sciences* 60.3 (2000), pp. 592–629.

[57] Y. Gertner and T. Malkin. *Efficient Distributed (n choose 1) Oblivious Transfer*. Tech. rep. MIT-LCS-TR-714. MIT Lab of Computer Science, 1997.

[58] H. Ghodosi. 'Analysis of an Unconditionally Secure Distributed Oblivious Transfer'. In: *Journal of Cryptology* 26.1 (2013), pp. 75–79.

[59] H. Ghodosi. 'On insecurity of Naor-Pinkas' distributed oblivious transfer'. In: *Information Processing Letters* 104.5 (2007), pp. 179–182.

[60] O. Goldreich, S. Micali, and A. Wigderson. 'How to play any mental game'. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing - STOC'87*. New York, NY, USA: ACM, 1987, pp. 218–229.

[61] O. Goldreich and R. Vainish. 'How to Solve any Protocol Problem - An Efficiency Improvement (Extended Abstract)'. In: *Advances in Cryptology - CRYPTO'87*. Ed. by C. Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1988, pp. 73–86.

[62] S. Goldwasser and L. Levin. 'Fair Computation of General Functions in Presence of Immoral Majority'. In: *Advances in Cryptology - CRYPTO'90*. Ed. by A. J. Menezes and S. A. Vanstone. Vol. 537. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1991, pp. 77–93.

[63] Y. Ishai and E. Kushilevitz. 'Improved Upper Bounds on Information-theoretic Private Information Retrieval (Extended Abstract)'. In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing - STOC'99*. New York, NY, USA: ACM, 1999, pp. 79–88.

[64] T. Itoh. 'Efficient Private Information Retrieval (Special Section on Cryptography and Information Security)'. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E82-A.1 (1999), pp. 11–20.

[65] S. Jiang, H. Li, and B. Li. 'Distributed Oblivious Transfer with Adaptive Queries'. In: *International Conference on Communications and Mobile Computing, Volume 1 - CMC 2010*. IEEE, 2010, pp. 213–217.

[66] F. W. Kasiski. *Die Geheimschriften und die Dechiffrir-Kunst*. Berlin: Mittler & Sohn, 1863.

[67] A. Kerckhoffs. 'La cryptographie militaire (1re partie)'. In: *Journal des sciences militaires* IX (Jan. 1883), pp. 5–38.

[68] A. Kerckhoffs. 'La cryptographie militaire (2de partie)'. In: *Journal des sciences militaires* IX (Feb. 1883), pp. 161–191.

[69] J. Kilian. 'Founding Cryptography on Oblivious Transfer'. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing - STOC'88*. New York, NY, USA: ACM, 1988, pp. 20–31.

[70] L. Kissner and D. Song. 'Privacy-Preserving Set Operations'. In: *Advances in Cryptology - CRYPTO 2005*. Ed. by V. Shoup. Vol. 3621. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2005, pp. 241–257.

[71] C.-H. Li and J. Pieprzyk. 'Conference Key Agreement from Secret Sharing'. In: *Information Security and Privacy - ACISP'99*. Ed. by J. Pieprzyk, R. Safavi-Naini, and J. Seberry. Vol. 1587. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1999, pp. 64–76.

[72] H. Lipmaa. 'An Oblivious Transfer Protocol with Log-Squared Communication'. In: *Information Security - ISC 2005*. Ed. by J. Zhou, J. Lopez, R. H. Deng, and F. Bao. Vol. 3650. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2005, pp. 314–328.

[73] M. Naor and B. Pinkas. 'Computationally Secure Oblivious Transfer'. In: *Journal of Cryptology* 18.1 (2005), pp. 1–35.

[74] M. Naor and B. Pinkas. 'Distributed Oblivious Transfer'. In: *Advances in Cryptology - ASIACRYPT 2000*. Ed. by T. Okamoto. Vol. 1976. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2000, pp. 205–219.

[75] M. Naor and B. Pinkas. 'Efficient Oblivious Transfer Protocols'. In: *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms - SODA'01*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001, pp. 448–457.

[76] M. Naor and B. Pinkas. 'Oblivious Transfer and Polynomial Evaluation (Extended Abstract)'. In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing - STOC'99*. New York, NY, USA: ACM, 1999, pp. 245–254.

[77] M. Naor and B. Pinkas. 'Oblivious Transfer with Adaptive Queries'. In: *Advances in Cryptology - CRYPTO'99*. Ed. by M. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 1999, pp. 573–590.

[78] V. Nikov, S. Nikova, B. Preneel, and J. Vandewalle. 'On Unconditionally Secure Distributed Oblivious Transfer'. In: *Progress in Cryptology - INDOCRYPT 2002*. Ed. by A. J. Menezes and P. Sarkar. Vol. 2551. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2002, pp. 395–408.

[79] T. Nishide and K. Ohta. 'Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol'. In: *Public Key Cryptography - PKC 2007*. Ed. by T. Okamoto and X. Wang. Vol. 4450. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2007, pp. 343–360.

[80] M. O. Rabin. *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. Tech. rep. MIT-LCS-TR-212. Massachusetts Institute of Technology - Laboratory for Computer Science, 1979.

[81] M. O. Rabin. *How to Exchange Secrets with Oblivious Transfer*. Tech. rep. TR-81. Aiken Computation Lab., Harvard University, 1981.

[82] M. O. Rabin. 'Randomized byzantine generals'. In: *Foundations of Computer Science - FOCS 1983*. IEEE, 1983, pp. 403–409.

[83] T. Rabin. 'Robust Sharing of Secrets when the Dealer Is Honest or Cheating'. In: *Journal of the ACM* 41.6 (1994), pp. 1089–1109.

[84] T. Rabin and M. Ben-Or. 'Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract)'. In: *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing - STOC'89*. New York, NY, USA: ACM, 1989, pp. 73–85.

[85] I. S. Reed and G. Solomon. 'Polynomial codes over certain finite fields'. In: *Journal of the Society for Industrial & Applied Mathematics* 8.2 (1960), pp. 300–304.

[86] R. Rivest, A. Shamir, and L. Adleman. 'A Method for Obtaining Digital Signatures and Public-Key Cryptosystems'. In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[87] R. L. Rivest. 'Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer'. Unpublished manuscript. 1999.

[88] A. V. Sergienko, ed. *Quantum Communications and Cryptography*. CRC Press, 2006.

[89] A. Shamir. 'How to Share a Secret'. In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[90] C. E. Shannon. 'A Mathematical Theory of Information'. In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423.

[91] C. E. Shannon. 'A Mathematical Theory of Information (Concluded from July 1948 issue)'. In: *Bell System Technical Journal* 27.4 (Oct. 1948), pp. 623–656.

[92] C. E. Shannon. 'Communication Theory of Secrecy Systems'. In: *Bell System Technical Journal* 28.4 (1949), pp. 656–715.

[93] D. R. Stinson. 'An Explication of Secret Sharing Schemes'. In: *Designs, Codes and Cryptography* 2.4 (1992), pp. 357–390.

[94] D. R. Stinson. 'On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption'. In: *Designs, Codes and Cryptography* 12.3 (1997), pp. 215–243.

[95] D. R. Stinson and R. Wei. 'Unconditionally Secure Proactive Secret Sharing Scheme with Combinatorial Structures'. In: *Selected Areas in Cryptography - SAC'99*. Ed. by H. Heys and C. Adams. Vol. 1758. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2000, pp. 200–214.

[96] M. Tompa and H. Woll. 'How to Share a Secret with Cheaters'. In: *Journal of Cryptology* 1.3 (1989), pp. 133–138.

[97] S. J. Wiesner. 'Conjugate Coding'. In: *SIGACT News* 15.1 (1983), pp. 78–88.

[98] D. Woodruff and S. Yekhanin. 'A Geometric Approach to Information-Theoretic Private Information Retrieval'. In: *Proceedings of the Twentieth Annual IEEE Conference on Computational Complexity*. IEEE, 2005, pp. 275–284.

[99] A. C.-C. Yao. 'How to Generate and Exchange Secrets'. In: *Foundations of Computer Science - FOCS 1986*. IEEE, 1986, pp. 162–168.

[100] A. C.-C. Yao. 'Protocols for Secure Computations'. In: *Foundations of Computer Science - FOCS 1982*. IEEE, 1982, pp. 160–164.

[101] S. Zhong and Y. Richard Yang. 'Verifiable Distributed Oblivious Transfer and Mobile Agent Security'. In: *Proceedings of the 2003 Joint Workshop on Foundations of Mobile Computing*. DIALM-POMC'03. New York, NY, USA: ACM, 2003, pp. 12–21.

[102] S. Zhong and Y. Richard Yang. 'Verifiable Distributed Oblivious Transfer and Mobile Agent Security'. In: *Mobile Networks and Applications* 11.2 (2006), pp. 201–210.