JAMES COOK UNIVERSITY, TOWNSVILLE
College of Science and Engineering

Investigation of Unconditionally Secure Multi-Party Computation

by

Louis Cianciullo

A Thesis Submitted
in Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Townsville, Australia

Dissertation directed by Associate Professor Hossein Ghodosi
Discipline Information Technology

2022

Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I certify that all information sources and literature used are quoted in the thesis.

ABSTRACT

Under *Information Systems* in the Encyclopaedia Britannica it is stated that "individual privacy hinges on the right to control one's personal information." The very next sentence then moves on to say that "while invasion of privacy is generally perceived as an undesirable loss of autonomy, government and business organizations do need to collect data in order to facilitate administration and exploit sales and marketing opportunities."

In the same paragraph the importance of individual privacy is highlighted and then dismissed, citing the need for corporations and governments to store and make use of an individual's private information. How can one reconcile the difference in these two statements? What possible way forward is there that allows for both the privacy of the individual and the collection and use of the individual's information by external (and potentially untrusted) parties. The field of cryptography and more specifically, the field of multi-party computation (MPC), holds answers to this problem.

MPC is a cryptographic protocol that allows for computation upon private information. Using MPC a set of parties can compute any given function (ranging from a simple summation, to an in depth statistical analysis) across private information, without ever actually revealing said information. This thesis is concerned with the study and exploration of MPC and two related protocols, secret sharing and oblivious polynomial evaluation (OPE).

Secret sharing is a fundamental cryptographic primitive that acts as a building-block for MPC. A secret sharing scheme involves a dealer who distributes a secret among a set of of participants. Each of these participants is assigned a share of the secret by the dealer. At a later date, qualified subsets of these participants are able to pool their shares together and reconstruct the secret; unqualified subsets of participants cannot gain any information relating to the secret. Secret sharing is a well established privacy preserving tool, whilst OPE has only been established

relatively recently.

An OPE scheme involves two parties, a sender, who holds a secret polynomial, $f(x)$, and a receiver who wishes to know an evaluation on the sender's polynomial, $f(\alpha)$. OPE allows the receiver to learn their desired evaluation without giving away the evaluation point, α, whilst also ensuring that the receiver cannot learn any other information relating to the sender's polynomial, $f(x)$. This protocol seems somewhat more niche on initial inspection (in comparison to secret sharing and MPC), however recently published results indicate that OPE is an important protocol in its own rights, particularly as a building-block for MPC.

In this thesis each of these three privacy-related areas is investigated, with the main goal being the establishment of new and more efficient or secure protocols. The specific results are summarised as follows:

Secret Sharing

1. *Improvements to Almost Optimum Secret Sharing with Cheating Detection:* The flaws in two existing secret sharing scheme that claim to achieve security against participants who lie or submit modified shares (AKA cheaters) are identified and summarily fixed. The resulting, fixed, schemes are relatively efficient and use a unique sub-protocol to gain security.

2. *Secret Sharing with Outsourced Cheater Detection:* A new mechanism is developed that allows for a set of external parties, i.e. not the participants, to examine the validity of a secret before it is actually reconstructed. The major benefits of the resulting 'outsourced' secret sharing scheme are that, not only is cheating detected with high probability, but also, upon cheating being detected, the actual cheaters themselves now learn nothing about the secret, something which was not always the case in previous results.

Oblivious Polynomial Evaluation

1. *Distributed Oblivious Polynomial Evaluation:* The notion of a distributed OPE scheme (DOPE) is formalised as an OPE in which the function of the sender is

distributed among a set of special servers. Along with this formal definition, an efficient and secure DOPE protocol is also devised. DOPE allows for greater privacy for the sender, as well as a higher level of reliability for both parties; as the sender does not actually have to be present for the execution of the OPE, allowing the receiver to complete the protocol at their leisure.

2. *Unconditionally Secure OPE: A Survey and New Results:* The existing unconditionally (information theoretic) secure OPE schemes are examined and formally categorised. Multiple extensions are proposed to improve upon both the efficiency and security of existing schemes; the flaws in a previously published OPE scheme are also identified.

Multi-Party Computation

1. *Efficient Information Theoretic MPC from OPE:* Utilising both secret sharing and OPE, a unique MPC scheme is devised. This new MPC scheme is extremely efficient for certain types of functions and presents an interesting new method for computing multiplications in MPC.

2. *Maliciously Secure OPE:* Taking the lessons learnt in the previous research on detecting cheaters in secret sharing, the MPC scheme devised above is improved upon to tolerate outright cheating and dishonesty from a majority of participants. The resulting scheme adds minimal overhead to the existing MPC.

Each of the results summarised above are based on published or submitted peer-reviewed articles that seek to incrementally improve upon the field of MPC and privacy related research. In short, this thesis looks to provide new tools and possibilities for preserving one's privacy and data by analysing and identifying flaws in the current research, as well as producing entirely new protocols.

Dedication

To Ness.

Acknowledgements

First and foremost, many thanks to Hossein for his patience and invaluable guidance. Your careful support and assistance has helped add a level of polish to my writing I never would have achieved by myself.

A huge thank you to Mangalam, I wouldn't have got this far without your help, thank you for the many pep talks and the research/writing group you organised.

To Mum and Dad, thank you for always encouraging me and egging me on (even if you didn't quite understand what all this 'secret squirrel' business was about), I look forward to wearing the 'Mario hat' at graduation.

To my friends, thanks for the many de-stressful (and certainly not wasted) nights of playing computer games, they kept me sane.

Last, and by no means least, I want to thank my wife, Vanessa. It's been a long time coming, thank you for being there with me every (stressful) step of the way. Thank you for celebrating every small victory and published paper with me and (just as importantly) cheering me up after every rejected paper and broken proof.

Louis Cianciullo

Adelaide, Australia, 2022.

List of Publications and Contribution of Others

Each of the following publications correspond to a specific chapter of this thesis. The author contributed to the bulk of this research, with help and guidance from associate professor Hossein Ghodosi.

1. **Chapter 2 - L. Cianciullo**, and H. Ghodosi, "Improvements to Almost Optimum Secret Sharing with Cheating Detection." *International Workshop on Security (Springer LNCS)*, pp. 193-205, 2018.
2. **Chapter 3 - L. Cianciullo**, and H. Ghodosi, "Outsourced Cheating Detection for Secret Sharing" *International Journal of Information Security, Springer*, pp. 1-8, 2021.
3. **Chapter 4 - L. Cianciullo**, and H. Ghodosi, "Unconditionally secure distributed oblivious polynomial evaluation." *International Conference on Information Security and Cryptology (Springer LNCS)*, pp. 132-242, 2018.
4. **Chapter 5 - L. Cianciullo**, and H. Ghodosi, "Unconditionally Secure Oblivious Polynomial Evaluation: A Survey and New Results." *Journal of Computer Science and Technology, Springer*, pp. 443-458, 2022.
5. **Chapter 6 - L. Cianciullo**, and H. Ghodosi, "Efficient Information Theoretic Multi-party Computation from Oblivious Linear Evaluation." *IFIP International Conference on Information Security Theory and Practice (Springer LNCS)*, pp. 78-90, 2018.
6. **Chapter 7 - L. Cianciullo**, and H. Ghodosi, "OLE-Based MPC Secure Against a Malicious Adversary" ***Draft***, 2022.

The research carried out in this thesis was supported by an Australian Government Research Training Program (RTP) Scholarship.

Contents

Chapter 1

Introduction

Once private information has been revealed, whether personal or business related, it can never be taken back. This is particularly true in today's interconnected society. It is therefore imperative that privacy preserving tools exist that allow individuals and organisations to not only statically secure their data (i.e., protect it at rest), but also allow them to freely make use of it.

As an example, consider the case outlined in [66], wherein a set of nations each have numerous satellites in orbit around the earth. Each of these nations wish to avoid collisions with another nation's satellites, without having to actually reveal the orbital paths of their own satellites. The naive solution to this problem is to simply have all of the nations hand over their information to a neutral third party, who can then compute and warn respective nations if a collision is likely. There is, of course, an obvious problem with this solution. It depends entirely on the trustworthiness of the third party. So how can we solve this issue without relying on a third party? Enter multi-party computation (MPC).

MPC is a cryptographic protocol that allows a set of n participants, $P_1, \cdots, P_n$, with private information (respectively), $x_1, \cdots, x_n$, to compute any given function $f(x_1, \cdots x_n)$, without explicitly revealing their private information. In simpler terms, the respective nations mentioned in the example above, could carry out an MPC protocol to compute a set of potential satellite collisions, without explicitly revealing the orbital paths of their satellites. The only information a given nation would obtain, regarding another nation's satellite, is what is explicitly revealed by the

computed collisions, i.e., what the evaluation of the function $f(x_1, \cdots x_n)$ reveals.

The beauty of an MPC protocol is that the 'function' (in our case the computing of collisions) need not be known in advanced. They are generalised protocols that can be used to privately and securely compute any given function. In fact, as the reader will see, there are even MPC protocols that can tolerate a set of dishonest or cheating participants, who deliberately spread misinformation. Specific MPC protocols also exist that will allow a set of participants to offload the entire computation to a set of third parties or servers, without letting this set of third parties learn anything [92]. Since its inception in the early 1980s [101], MPC has been extensively researched within the cryptographic community, as a generalised solution to any scenario in which computation must be carried out on private data.

A complementary field of research to MPC is the field of privacy preserving protocols. Within this thesis, a privacy preserving protocol can be seen as a sort of specialised MPC protocol that only computes a specific function (see [22, 32, 76] for some examples). The usual benefit of a privacy preserving protocol is greater communication and/or information efficiency. Drawing back to our previous example, a privacy preserving protocol specifically tailored towards the computation of computing collisions in a set of paths may perform significantly faster for our nations than a generalised MPC protocol that can solve a variety of different tasks.

The benefits of each approach is evident, on one hand we can potentially save valuable time and resources using a dedicated privacy preserving protocol, on the other, we lose the generality and adaptability of an MPC protocol. In general, if the function is highly specialised or complicated and/or is known in advanced a privacy preserving protocol will probably be suitable, for all other cases MPC reigns supreme.

In this thesis both MPC and privacy preserving protocols are investigated. In

actual fact the privacy preserving protocols we investigate are actually also used as primitives and building blocks within MPC protocols. Specifically, this thesis can be broken up into a set of three parts, each containing two Chapters. In Part I we investigate the protocol known as secret sharing. Part II is concerned with a more recently discovered protocol called oblivious polynomial evaluation (OPE). Finally, Part III describes an MPC protocol that actually makes use of both secret sharing and OPE. The main focus of this work is the designing of efficient and secure protocols within each of these respective and inter-related fields.

In the next section we shed some light on what each of these fields entail and informally describe and give some brief history on the aforementioned areas of study. In the name of brevity and clarity we have elected to spare the reader and introduce much of the more technical detail in each of the following Chapters as required. As such, what follows is not a laborious set of definitions and technical theorems, but rather, what we hope, is an approachable introduction to the cryptographic protocols of secret sharing, OPE and MPC.

1.1 Background

The first step to designing or even describing a cryptographic protocol is to formally specify the operational environment the protocol is to be running in. For example, is all information public? Can participants communicate securely/privately with one another? How much computational power is available for each participant? What kind of adversary are we facing? And so and so forth. The formal term used for this description is a security model.

Each of the protocols designed or investigated within this thesis have a corresponding security model that is rigorously defined in conjunction with the actual protocol itself. So to avoid repetition, we will not individually describe the models used in each of the six Chapters of this thesis. Rather, we instead give a more

high level overview of a generalised security model that holds for all of the protocols described.

1.1.1 Security Model

The specific focus of this thesis is on information theoretic, threshold protocols in which we consider semi-honest or malicious adversaries. To unpack this statement we can start by defining what exactly is meant by an adversary.

The protocols described within this thesis are cryptographic protocols that involve a set of $n \geq 2$ participants. For each of these protocols we assume the presence of an adversary who is able to take control of, and gain all of the information held by, a subset of these participants. The type of adversaries we consider are widely accepted and researched within the literature [19], coming in two different flavours:

Semi-Honest (Passive) Adversary: The participants controlled by a semi-honest adversary will not deviate (i.e., cheat) from the protocol, however, they will privately communicate and pool their knowledge and resources in an attempt to learn or compute some extra information not explicitly assigned to them.

Malicious (Active) Adversary: This type of adversary is a step above a semi-honest adversary. Participants controlled by a malicious adversary will actually deviate from the protocol e.g., sending false information, lying, not sending information or tampering with held information. Again, this is done in an attempt to either disrupt the protocol or to gain extra information.

As the protocols we consider are all threshold protocols, the subset of participants controlled by the adversary is defined as some threshold amount $t < n$. To elaborate, a maliciously secure protocol in this model is secure against an adversary controlling up to $t < n$ participants who may outright lie e.g., sharing or supplying incorrect information. To draw back to the previous example regarding the orbital paths

of a nation's satellite; a malicious adversary who has seized control of one of the nations, could have said nation provide a completely false orbital path to the MPC, whilst a semi-honest adversary could not. We should note that the threshold value will change depending on the specific protocol we consider (for instance, it could be that $2t < n$), however in all cases t is less than n, meaning that even for the absolute worst case, to carry out the protocol we require at least one participant who is honest.

This takes care of the threshold and adversary statements made in the opening sentence for this sub-section. All that remains is to define what is meant by an information theoretic secure protocol. Essentially, an information theoretic secure protocol, or an unconditionally secure protocol, places no computational bounds on any of the participants (unlimited space, time and processing power). As a direct consequence of this, there is, of course, also no bounds placed on the adversary. This is in contrast to a computationally secure protocol which may make use of some hardness assumption that places bounds on a given participant's computational power.

For instance, if an efficient algorithm for determining the prime factors of a given value were found then the the security of the RSA encryption protocol [96] would be compromised. The security of the RSA protocol relies on the inability of an adversary to efficiently find specific prime factors of a large integer. Although this works well against a bounded adversary, who has limited time and computational power, finding these factors for an unbounded adversary is trivial, even using the simple brute-force approach of trying every possible combination of prime numbers. As a result of this, unconditionally secure protocols are not based on any hardness conjectures and cannot be broken.

Aside from the obvious security benefits that information theoretic security has

over computational security, Halevi et al. [61] state that information theoretic protocols are "typically simpler and have better concrete communication and computation costs than their computational counterparts." In general the computational costs of an information theoretic protocol are much less than that of a computationally secure protocol. A major downside to unconditional security however, is that it is not possible to have an information theoretic two-party (i.e., $n = 2$) protocol [3, 28]. Furthermore a computationally secure protocol can achieve security against a majority of dishonest participants, whilst an information theoretic protocol requires a majority of participants to be honest. This limitation can however, be overcome by utilising an external source of information or making other such assumptions (as we will see in Chapters 6 and 7).

In fact, in recent years a spate of research has been carried out on efficient information theoretic MPC protocols [48, 47, 79]. These results are not merely theoretical, with many protocols being suitable for implementation [72, 73]. It is for these reasons that, within the bounds of this thesis, we choose to focus on information theoretic protocols.

From a historical perspective, the first of the different types of protocols discussed in this thesis is that of secret sharing; which was discovered in 1979 [11, 97]. Following this is MPC, which was originally discussed by Yao [101] for the two party case and then later, generalised to n participants in the information theoretic setting by Goldreich et al. [60] and Ben-Or et al. [7]. Lastly OPE was discovered in 1999 by Naor and Pinkas [81]. Given this, it seems only fitting that we present the background information on each of the three protocols in chronological order of discovery.

1.1.2 Secret Sharing

Secret sharing is a cryptographic protocol in which a secret, S, is divided into a set of shares, $V_1, \cdots, V_n$, and distributed to n participants, $P_1, \cdots, P_n$, where P_i gets the share V_i for $i = 1, \cdots, n$. The (t, n) threshold case, which is considered here, is such that $t < n$ or less cooperating participants cannot gain any information about the secret, whilst $t + 1$ or more participants can pool their shares and reconstruct the secret. The general gist of the protocol is that a special participant named a dealer ($\mathcal{D}$) divides the secret among the participants, by privately assigning them shares. At a later date $t + 1$ or more participants pool their shares and perform a computation to reconstruct S.

Secret sharing was first discovered independently by Blakley [11] and Shamir [97] in 1979. The classic use case given by Shamir is to utilise secret sharing to safeguard cryptographic keys, by distributing the key among participants via a threshold secret sharing scheme. In particular, Shamir argues that a very robust key management scheme can be achieved when the set of participants is equal to $n = 2t + 1$. The reason being that the key can be recovered even if $\lfloor n/2 \rfloor$ of the original pieces (or shares) are destroyed.

As it turns out, secret sharing has far more uses than merely protecting data, passwords or keys at rest. In fact, it was through secret sharing that the first MPC schemes were achieved [7, 60]. The literature spanning the field of secret sharing is exhaustive, with various different modifications and additions taken into consideration. However, the main drive of research in this field (that we focus on) is to construct efficient protocols (i.e., small share size and low communication) that better handle potential cheaters, AKA a malicious adversary.

The first to consider cheating in secret sharing was Tompa and Woll [99]. They considered the case in which a participant does not follow the protocol, but instead

lies to the other participants at reconstruction time and submits false information in place of their real share. They proved that just a single cheater can not only prevent all of the honest participants from recovering the secret, but also easily learn the secret themselves.

To prevent this scenario, Tompa and Woll devised a modified secret sharing scheme that allowed participants to detect cheating; henceforth we refer to such a scheme as a secret sharing scheme with cheater detection (SSCD). Their original SSCD scheme, however, was not very efficient, having a large share size (the amount of information each participant has to hold). Thankfully, in the years following Tompa and Woll's work, many efficient SSCD protocols were discovered, along with several different variants of cheating resistant secret sharing schemes, which are briefly examined below.

Secret Sharing with Cheater Detection (SSCD)

SSCD considers the case whereby participants wish to simply detect if cheating has occurred. This is accomplished by utilising a normal secret sharing scheme and also having the dealer distribute some extra information among participants. This extra information is utilised at reconstruction time by a pool of $t + 1$ participants to identify if the reconstructed secret is valid or not (i.e., if cheating has occurred). If cheating has been detected then any honest participants know not to accept the reconstructed secret as valid. An SSCD scheme requires at least one out of the $t+1$ participants who are reconstructing the secret to be honest.

Because of its limited nature, SSCD is by far the most efficient of the cheater resistant secret sharing protocols (in terms of both share size and communication complexity), additionally it has been shown that SSCD can act as a building block in more complex protocols [40].

Secret Sharing with Cheater Identification (SSCI)

SSCI performs in much the same way as SSCD, in that the basic premise is a secret sharing scheme in which the dealer has assigned some extra information to participants. As with SSCD, a set of $t + 1$ participants perform a reconstruction computation and, during this reconstruction phase, utilise their extra information in order to detect cheating. In SSCI, however, participants also utilise their extra information to actually identify who the cheaters are. The downside to this extra functionality is that SSCI requires that there can only be, at most, $k < (t + 1)/2$ cheaters [85]. Put simply, in an SSCI scheme we require a majority of honest participants, in order to actually identify cheaters.

Robust Secret Sharing

Robust secret sharing begins in much the same way as SSCI and SSCD, with the dealer privately distributing some extra information alongside each participant's share. However, the goal of a robust secret sharing scheme is to actually recover the secret, even if some of the participant's have cheated. A limitation of this protocol is that at least $2t + 1 \leq n$ participants are required for the reconstruction phase, only t of which can be cheaters [10]. Furthermore the reconstruction algorithm of a robust secret sharing scheme is far more complex than both SSCD and SSCI [21].

Verifiable Secret Sharing (VSS)

The final type of secret sharing scheme we consider is VSS. This protocol was originally discovered by Chor [27] for the computationally secure setting, and then later defined for the information theoretic setting in the late 1980s [7, 60, 94]. This type of secret sharing scheme can be seen as an extended robust secret sharing scheme, with the additional assumption that even the dealer could actually be a cheater.

This extra assumption came about due to VSS's use as a key building block in the early days of MPC. Researchers needed a tool that would allow participants to distribute private information amongst each other, in such a way that the private information, as well as the shares of the private information, can all be easily verified (i.e., check shares have not been tampered with and ensure the dealer has actually distributed the shares correctly).

VSS schemes are complex protocols with reconstruction/verification phases that consist of multiple rounds of communication [94]. As with the other secret sharing schemes, research within this field focuses on reducing share size; additionally a great deal of research has gone into actually reducing the rounds of communication needed in VSS [57].

This is easily the most intensive and complex of the four cheater resistant protocols listed, mainly due to the fact that VSS is used within MPC to force participants to commit to a particular value and provide proof to the other participants that they have in fact been truthful. However, of late VSS have been usurped as the favoured tool or building block in MPC [9]. Instead researchers are leaning towards the approach used in the lighter SSCD schemes, utilising what is known as message authentication codes (MACs) [9, 94], to achieve similar results in both efficiency and security as SSCD schemes.

Focus of Research in Secret Sharing

In light of the above, this thesis' main focus is on SSCD schemes. We further examine this protocol in Chapters 2 and 3, respectively showing the flaws in and then fixing a previously defined SSCD scheme, and establishing a new type of SSCD scheme in which reconstruction of the secret is outsourced to a set of special servers. As previously mentioned, secret sharing is used as a primitive in various privacy preserving and MPC protocols. The very first information theoretic multi-party

schemes that considered $n > 2$ participants [7, 60] utilised Shamir's secret sharing scheme [97] as a core building block, allowing participants to securely distribute information amongst each other, in the form of shares. This is further discussed below.

1.1.3 MPC

MPC was first conceived by Yao [101] in 1982, who looked at utilising what is known as garbled circuits, to allow two parties to securely compute a function across their private inputs, without explicitly revealing their inputs. Yao's result was later generalised by Goldreich et al. [60] and Ben-Or et al. [7] who both developed n party information theoretic MPC protocols secure against up to $t < n/3$ participants. Roughly two or so years after Ben-Or et al.'s result, Rabin et al. [94] achieved information theoretic MPC secure against $t < n/2$ participants, by allowing a small probability of error.

The general approach these early results had each participant utilise a VSS scheme to distribute their information among the other participants in the form of shares. Linear operations could then be carried out privately by just performing the corresponding operation (e.g., addition between two input values) on the shares of each participant's input.

Multiplication, however, requires a prohibitive amount of communication [7], particularly in the malicious setting, where participants must use a VSS each time they wish to distribute information to the other parties. In light of these facts, the bottleneck in efficiency of these early MPC results can be traced to 1) the prohibitive (communication) cost associated with computing multiplication and 2) the additional, further cost to communication acquired when using a VSS scheme to validate each participant's shared information.

The first of these inefficiencies was solved by Beaver [4] in his commodity model,

who provided a means of fast multiplication by splitting the MPC protocol into two phases:

1. **Preprocessing Phase:** In this phase participants compute or are given some random, but correlated, data i.e., shared data that has nothing to do with any of their private information.

2. **Online Phase:** This phase is where the actual MPC takes place. Participants utilise the shared data to efficiently compute multiplication in MPC.

The main benefit to this approach is that the bulk of the computation can be delegated to the preprocessing phase, which can be completed at the participant's leisure (as it is independent of their private inputs). This allows for an extremely efficient online phase, letting participants efficiently perform the required MPC.

The second of these inefficiencies has been solved in recent times, by a variety of different researchers [9, 48, 72]. The main gist of these results being a general lowering of the high security expectations laid out in the early results of Goldreich et al. [60] and Ben-Or et al [7]. Specifically, where these early results relied on full-blown VSS protocols that could be used to not only detect, but also fix any cheating that occurred, the more modern results instead rely on simple MAC schemes that are extremely similar to SSCD protocols (simply detecting cheating). Of course, the downside to this transition in security models is that cheating can only be detected, not fixed, nor can the cheaters be identified. However, the efficiency boosts realised from this paradigm shift are undeniable, with practical MPC implementations starting to take their place in the real world as vital privacy preserving tools [16].

1.1.4 Oblivious Polynomial Evaluation

OPE is a relatively new cryptographic protocol that has roots in another, older protocol, know as oblivious transfer (OT). An OT protocol is a two party protocol that involves a sender, who holds a set of secret values, and a receiver who wishes to learn one of these secrets. OT allows the receiver to learn just one of the sender's secrets without actually letting the sender know which secret was learned.

Since its inception [56, 93], OT has been heavily researched and used extensively as a key building block in MPC and privacy preserving protocols [69, 72]. OPE is closely related to this protocol, also being composed of a sender and a receiver. The difference, however, is that in an OPE protocol, the sender holds a polynomial, $f(x)$, and the receiver wishes to learn a specific evaluation of this polynomial, $f(\alpha)$. Security and privacy are maintained in an OPE protocol if the receiver only gains information relating to their evaluation (i.e., they do not learn anything further about the sender's polynomial) and if the sender cannot learn anything about the receiver's evaluation point (α).

In contrast to OT, OPE is a relatively new protocol, having been discovered by Naor and Pinkas [81] in 1999 and then formalised and refined in 2006 [83]. OPE has not been as heavily researched as OT, however, of late there has been an emphasis on utilising OPE in MPC and privacy preserving protocols [22, 31, 53], in much the same fashion as OT is used; as a vital building block. A key example of this is given in Chapters 6 and 7 of this thesis, in which an MPC scheme is designed around an efficient OPE protocol.

Our research on information theoretic OPE is described in Chapters 4 and 5, which look at, respectively:

1. Developing an efficient distributed OPE scheme, in which a set of servers handle the responsibilities of the sender.

2. Categorising and performing a thorough investigation of the existing information theoretic OPE schemes within the literature, along with multiple extensions and improvements to these existing protocols.

As with all fields and protocols thus far summarised, we leave the technical details and definitions for the corresponding Chapters (in this case Chapters 4 and 5).

1.2 Motivation

MPC has applications in a vast variety of real-world scenarios such as secure auctions, electronic voting, privacy preserving machine learning and statistics, private information retrieval and threshold cryptography [55]. However, it is only recently that MPC has become practical enough for implementation, as such, it is vital that research continues to push forward in this area, in order to provide privacy and control of data for everyone. By improving upon already existing protocols, as well as designing newer and more efficient MPC and privacy preserving protocols, we hope to help move incrementally closer to these goals.

Of particular note at the time of writing, is the ongoing COVID 19 pandemic overtaking the world. A common solution to this problem is the introduction of contact tracing mobile applications that not only record a user's location, but also who else they have been in contact with [33]. Whilst these applications may indeed save lives, they are also relying on the divulgement of very specific and private locational and behavioural data. The backlash that would occur in the event of this data being misused or stolen would be extensive. Thus, it is vital that proven privacy preserving solutions, such as MPC, are available and efficient enough to be implemented.

1.3 Outline

Each of the Chapters within this thesis is based on (or builds on the work carried out in) a published, peer reviewed conference or journal article. As such they have been written in a self-contained fashion wherein all of the necessary definitions and background is provided. It is for this reason that we do not include a preliminary or dedicated background section.

Chapters 2 and 3 examine SSCD, with Chapter 2 proving and then fixing the flaws in an already published SSCD scheme, and Chapter 3 going along a different path, by building a unique SSCD protocol in which the reconstruction and verification of the secret is not handled by the actual participants.

The next field examined is OPE. In Chapter 4 we formalise the notion of distributed OPE and construct an optimal scheme fitting this definition. Chapter 5 takes a step back and thoroughly examines the existing information theoretic OPE schemes, categorising each scheme as well as adapting and improving upon the existing literature and, lastly, showing that a previously published scheme is not secure.

The penultimate two Chapters of this thesis deal with MPC; in Chapter 6 we build an efficient MPC scheme secure against semi-honest adversaries. This scheme is the culmination of the work done in previous sections, combining both secret sharing and OPE. In Chapter 7 we improve upon our scheme, utilising an SSCD technique (MACs) to achieve security against a malicious adversary, at very little cost to computational or communication complexity. Finally, Chapter 8 concludes the thesis and establishes avenues of further research.

Chapter 2

Improvements to Almost Optimum Secret Sharing with Cheating Detection

2.1 Introduction

As previously remarked upon, this Chapter is concerned with secret sharing with cheating detection capability (SSCD), which allows participants to detect the submission of faulty or modified shares. Within this field researchers actually consider two different models of security, the OKS model [89] and the CDV model [20]. In this chapter we review both of these models and then demonstrate that two previously discovered SSCD schemes (one set in either model) fail to achieve security. After proving the deficiencies in these schemes we then show that with some modifications both schemes can be made secure.

The resulting, 'fixed' schemes have near optimal share size, support operations from an arbitrary finite field and provide a high level of security even if the secret domain is small. The first of our fixed schemes is devised under the OKS model and is the most efficient of its kind, whilst the second is devised under the CDV model and is as efficient as the current best solution. Before launching into specifics, we present, in more formal and greater detail, some background information on SSCD.

2.1.1 Background

In a secret sharing scheme a secret, S, is distributed amongst n participants, $P_1, \cdots, P_n$, in the form of shares, $V_1, \cdots, V_n$, such that P_i obtains V_i for $1 \leq i \leq n$. At a latter date an authorised subset of participants is able to reconstruct S by

combining their shares. We can classify authorised and unauthorised subsets of participants by means of an access structure Γ, such that the subset $\mathcal{A} \in \Gamma$ is an authorised subset that can reconstruct S, whilst $A' \notin \Gamma$ is an unauthorised subset that cannot. In a perfect, unconditionally secure secret sharing scheme, if an unauthorised subset attempt to reconstruct S then they should obtain no additional information regarding S.

A secret sharing scheme is more formally defined as a pair of algorithms `SHARE` and `REC`. The `SHARE` algorithm is executed by a trusted entity $\mathcal{D}$, known as the dealer and `REC` is executed by a second, separate entity $\mathcal{C}$, the combiner. We define these algorithms in the following manner:

`SHARE`$(S) \rightarrow (V_1, \ldots, V_n)$: A probabilistic algorithm that takes a secret S and produces n random shares.

`REC`$(\gamma) \rightarrow S'$: A deterministic algorithm that takes a subset of shares γ and outputs a secret S'.

Given a secret $S \in \mathcal{S}$ that is used to compute shares $V_1, \cdots, V_n$ and a subset of these shares, denoted by γ, used to compute a secret S' then the following properties will hold (assuming that all participants follow the protocol exactly):

$$\Pr[S' = S \mid \gamma \in \Gamma] = 1$$

$$Pr[S' = S \mid \gamma \notin \Gamma] = Pr[S' = S]$$

These properties state that in a perfect secret sharing scheme a set of unauthorised participants cannot reduce their uncertainty of the secret. It is a well known fact that in a perfect secret sharing scheme the share size cannot be smaller than the secret itself. That is, $|V_i| \geq |S|$ where $|V_i|$ denotes the maximum size of a given participant's share i.e., $\max_{1 \leq i \leq n}(|V_i|)$.

As with all protocols examined in this thesis, we consider the threshold case, wherein a (t, n) threshold secret sharing defines an unauthorised subset of participants as any subset γ' in which $|\gamma'| < t+1$ where $t+1 \leq n$. Secret sharing schemes are an effective way to safeguard privacy, however due to the work of Tompa and Woll [99], it is well known that just one cheating participant can compromise the entire protocol. To protect against this, we can, of course, utilise SSCD.

To reiterate, SSCD considers the scenario in which corrupt participants, known as cheaters, modify their shares in order to trick other (honest) participants into reconstructing a false secret. An effective SSCD scheme can alert the honest participants to the submission of false shares. This type of protocol has applications in such things as robust secret sharing [40] and secure message transmission [74].

More formally a (t, n, δ) SSCD scheme is a threshold secret sharing scheme in which the probability of successful cheating occurring is less than or equal to δ [1]. To put this another way a (t, n, δ) SSCD scheme is a secret sharing scheme with a modified `REC` algorithm. In the definition that we consider, there are n participants, $P_1, \cdots, P_n$ with shares, $V_1, \cdots, V_n$ where $t+1$ of these same participants will attempt to reconstruct the secret S. We assume that up to t of these participants are dishonest (cheaters) and wish to force the honest participants into accepting a secret S', where $S' \neq S$.

The goal of SSCD is to detect when such cheating occurs in order to prevent reconstruction of an invalid secret. Thus, the `REC` algorithm given for secret sharing is modified so that it either outputs S' (which may or may not be equal to the original secret) or a special symbol, $\perp$, which indicates that cheating has been detected.

For any such scheme with the above reconstruction protocol in which up to t faulty shares, $V'_{i_1}, \cdots, V'_{i_t}$ and one unmodified share, $V_{i_{t+1}}$ are submitted, the prob-

ability of failure, δ, is defined as:

$$\Pr[\texttt{REC}(V'_{i_1}, \cdots, V'_{i_t}, V_{i_{t+1}}) = \perp] \geq 1 - \delta$$

$$\forall\, (S, V_1, \cdots, V_n) \text{ where } \exists\, V'_{i_j} \neq V_{i_j} \text{ for } j \in [1, n]$$

Where a faulty share is defined as a share that is different in value from the original share distributed by the `SHARE` algorithm.

Conversely let S be a valid secret distributed to a set of n participants by `SHARE` and define S' as the secret computed by `REC` using shares (which may or may not be modified, i.e., faulty) from $t+1$ of these same participants, then:

$$\Pr[S' \neq \perp | S' \neq S] \leq \delta$$

Which holds for all $V'_{i_1}, \cdots, V'_{i_t}$ where, as before, there exists at least one $V'_{i_j} \neq V_{i_j}$ for $j \in [1, n]$.

An extensive amount of work has been produced on SSCD with the main goal being the reduction of share size. A key factor that greatly influences share size is the model of security a SSCD scheme is devised under. Typically schemes are devised under one of two models, the CDV model and the OKS model.

2.1.2 CDV Model

The CDV model was given by Carpentieri, De Santis and Vaccaro [20] in 1994. Schemes devised under this model consider the scenario in which the cheaters actually know the secret. Thus, such a scheme is only secure if the honest participants can detect cheating (with probability $1 - \delta$) given that the cheaters already know the secret.

Ogata, Kurosawa and Stinson [89] give a bound for the share size of schemes devised under this model. As before let $|V_i|$ denote the maximum share size of a given participant. Let $|\mathcal{S}|$ denote the size of the secret domain and δ the probability of successful cheating occurring:

$$|V_i| \geq \frac{|\mathcal{S}| - 1}{\delta^2} + 1$$

2.1.3 OKS Model

In [89] Ogata, Kurosawa and Stinson introduced the OKS model. This model of security assumes that the cheaters do not know the secret, i.e., the secret is assumed to be uniformly random in the secret space. As a result of this the bound on share size for schemes developed under this model varies from the bound given by the CDV model:

$$|V_i| \geq \frac{|\mathcal{S}| - 1}{\delta} + 1$$

Remark 1. *SSCD schemes devised under the OKS model typically have smaller share sizes than those devised under the CDV model. However, a common trait for schemes developed under the OKS model is that δ is dependent on $|\mathcal{S}|$ e.g., $\delta = \frac{1}{\sqrt{|\mathcal{S}|}}$ [87]. This is not the case for schemes devised under the CDV model as in these type of schemes the probability of successful cheating does not necessarily depend on the secret domain.*

2.1.4 Our Contribution

First we present some flaws that exist in two SSCD schemes given in [71]. One of these schemes is developed under the OKS model and the other is developed under the CDV model. The flaws in question allow just one cheating participant to fool the other participants into accepting an incorrect secret. Secondly we show how to modify the schemes in [71], such that they are secure and satisfy the following desirable properties (of "good" protocols) originally given in [86] by Obana and Tsuchida:

- **Capable of supporting an arbitrary finite field:** Computations can be done in a field of any characteristic.

- **Near optimal share size:** The share size is only slightly larger than the bound given for the particular model the scheme is developed under, where an optimal scheme is one that meets the bound.

- **Adequate level of security even if the secret domain is relatively small:** The probability of successful cheating occurring is no larger than $\frac{1}{|\mathcal{S}|}$ or can be set arbitrarily (as is the case for most schemes developed under the CDV model).

Whilst most schemes developed under the CDV model have these attributes there is only one other scheme developed under the OKS model that currently achieves this, and it is given by Obana and Tsuchida in [86]. The share size for this scheme is 2 bits larger than optimal whilst our scheme developed under the OKS model has a share size only 1 bit larger than optimal. Making our scheme is the most efficient yet.

The other scheme presented in this Chapter is developed under the CDV model and achieves the same share size as the scheme presented by Cabello et al. [17]. To the best of our knowledge this scheme has the smallest share size of any such scheme secure in the CDV model.

2.2 Preliminaries

2.2.1 Shamir's Secret Sharing Scheme

In Shamir's seminal paper [97] he introduced a (t, n) threshold secret sharing scheme in which each participant is given a point on a polynomial of degree t for their share. Suppose that there are n participants, $P_1, \cdots P_n$ and that all computations are done in the field $\mathbb{F}_q$, where q is a prime number such that $q > n$. The protocol is as follows:

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$: A probabilistic algorithm in which $\mathcal{D}$ picks a random polynomial, $f(x)$ of degree at most t, where $f(0) = S$. Participants are assigned the share $V_i = f(i)$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow S'$: A deterministic algorithm that takes a subset of shares γ, where $|\gamma| \geq t+1$ and computes and outputs S' using Lagrange interpolation. Without loss of generality assume γ is composed of shares from the first set of $|\gamma|$ participants i.e., $P_1, \cdots, P_{|\gamma|}$, then:

$$S' = \sum_{i=1}^{|\gamma|} V_i \prod_{\substack{1 \leq i \leq |\gamma| \\ i \neq j}} \frac{j}{j-i}$$

In Shamir's scheme it is assumed that all participants follow the protocol exactly, i.e., they always submit the correct shares for REC, and this, $S' = S$.

2.2.2 The Tompa and Woll Attack

Tompa and Woll [99] showed that Shamir's scheme is vulnerable to an attack in which malicious participants are able to obtain the secret S and force the honest participants into reconstructing $S' \neq S$. In fact, it is possible for just one cheater in a group of $t+1$ participants to accomplish this. To carry out this attack successfully cheaters modify their shares in the following fashion.

Given a set of $t+1$ participants $P_1, \cdots, P_{t+1}$ attempting to reconstruct the secret, assume that there is subset of dishonest participants within this set denoted by θ where $1 \leq |\theta| \leq t$. This set of cheaters compute a polynomial $\Delta(x)$ of degree at most t. They set $\Delta(i) = 0$ for $P_i \notin \theta$ (all of the honest participants) and $\Delta(0) = \beta$, which is an arbitrary value picked by the cheaters. $P_j \in \theta$ then submits the share $V_j + \Delta(j)$ where V_j is the share originally assigned to P_j. All dishonest participants submit these modified shares; whilst $P_i \notin \theta$ submits his unaltered share V_i (which can be viewed as $V_i + \Delta(i)$).

This results in the reconstruction of $f'(x) = f(x) + \Delta(x)$ where $f(x)$ is the original polynomial used to compute and distribute shares. It is now easy for the cheaters to compute S as $f'(0) = S' = S + \beta$.

2.2.3 Review of an SSCD Scheme Devised in OKS model

In this section the scheme developed under the OKS model presented in [71] is reviewed. All computations are done in the field $\mathbb{F}_q$ where $q > 2n$.

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $f(x)$, of degree at most $2t$, where $f(0) = S$. $\mathcal{D}$ also chooses $2n$ distinct public elements $\alpha_1, \cdots, \alpha_{2n}$. Each participant, P_i is given the share $V_i = (f(\alpha_i), f(\alpha_{i+n}))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow S'$: A subset, γ, of $t+1$ participants (of which up to t can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$ using Lagrange interpolation. If the degree of $f'(x)$ is $2t$ or less, output $s' = f'(0)$ as the secret, otherwise output $\perp$.

2.2.4 Review of an SSCD Scheme Devised in CDV model

Here we present the SSCD scheme devised under the CDV model presented in [71]. All computations are done in the field $\mathbb{F}_q$ where $q > 3n$.

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $f(x)$, of degree at most $3t$, where $f(0) = S$. $\mathcal{D}$ also chooses $3n$ distinct public points $\alpha_1, \cdots, \alpha_{3n}$. Each participant, P_i is given the share $V_i = (f(\alpha_i), f(\alpha_{i+n}), f(\alpha_{i+2n}))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow S'$: A subset, γ, of $t+1$ participants (of which up to t can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$ using Lagrange interpolation. If the degree of $f'(x)$ is $3t$ or less, output $s' = f'(0)$ as the secret, otherwise output $\perp$.

2.3 Flaws in The Reviewed SSCD Schemes

This section is concerned with the security flaw inherent in the two schemes presented by in [71] (henceforth refereed to as the JS schemes). It is shown that neither of these two schemes is secure. We begin by investigating the first scheme which was developed under the OKS model.

The authors state that this scheme will detect cheating with probability $\delta = 1 - \frac{1}{q}$ even if up to t of the $t+1$ participants are cheaters. The mechanism used for cheating detection relies on the fact that a degree $2t$ polynomial is uniquely defined by $2t+1$ points. So if $t+1$ participants submit their shares then $\mathcal{C}$ obtains $2t+2$ points. If these shares are unaltered then they will all describe the same degree $2t$ polynomial $f(x)$. However, if one or more of these shares are changed in a random fashion then, with probability $1 - \frac{1}{q}$, the collection of shares will describe a degree $2t+1$ polynomial.

For the above scenario in which shares are changed randomly the scheme is indeed secure. However, the scheme is not secure against the Tompa and Woll attack. Using this attack just one participant can cheat the other honest participants with probability 1, as demonstrated below.

Let θ denote the set of dishonest participants where $1 \leq |\theta| \leq t$. In order to cheat they must submit shares that are consistent with the honest participants' shares, i.e., all shares describe a polynomial of degree $2t$. To accomplish this they can use the Tompa and Woll attack to force reconstruction of $f'(x) \neq f(x)$, where $f'(x)$ is a polynomial of degree at most $2t$. To do so θ compute a polynomial $\Delta(x)$ of degree at most $2t$, such that for all honest participants $P_i \notin \theta$, $\Delta(\alpha_i) = \Delta(\alpha_{i+n}) = 0$ and $\Delta(0) = \beta$. All cheaters $P_j \in \theta$ submit the modified shares $V'_j = (f(\alpha_j) + \Delta(\alpha_j), f(\alpha_{j+n}) + \Delta(\alpha_{j+n}))$. This will result in the reconstruction of a polynomial $f'(x) = f(x) + \Delta(x)$ that is of degree at most $2t$ with $f'(0) =$

$S + \beta$. This occurs because of the well known (+, +) homomorphic property of Shamir's secret sharing scheme [8]. This property states that given two sets of shares, $r_1, \cdots, r_z$ and $k_1, \cdots, k_z$, associated with the polynomials $R(x)$ and $K(x)$ respectively, the individual summation of these shares: $r_1 + k_1, \cdots, r_z + k_z$, will describe the polynomial $R(x) + K(x)$.

The second of the JS schemes is devised under the CDV model. However, the cheating detection mechanism is the same as the first scheme described. This time however, $\mathcal{D}$ uses a polynomial of degree at most $3t$ to distribute shares and each participant is given three points on this polynomial as his share. Thus the exact same attack can be used against this scheme, with the cheaters computing a polynomial $\Delta(x)$ of degree $3t$.

2.4 Improvement to JS SSCD

In this section we show how to modify the JS SSCD schemes in order to add security. Both modified schemes are unconditionally secure with the same share size, secret space, and probability of successful cheating as claimed in the original protocols presented in [71]. The basic idea behind these corrected schemes is to use a slightly modified version of what Hoshino and Obana [67] call a check digit function, which is another name for a technique from the field of MPC that we've briefly touched on before, MAC tags.

In a cheater detection scheme that employs these check digit function, two types of shares are distributed: one directly related to the secret S, and the other related to a check digit $l = \mathcal{L}(S)$. At reconstruction of a secret S' and a check digit l', the combiner, $\mathcal{C}$ checks whether $l' = \mathcal{L}(S')$. If this is not the case then $\mathcal{C}$ concludes that cheating has occurred and outputs $\perp$, otherwise $\mathcal{C}$ assumes that S' is valid and broadcasts this value.

Our proposed scheme follows this basic definition. However, instead of constructing shares related to the check digit l and the secret S we instead distribute two or three shares (using the OKS or CDV models, respectively) to each participant corresponding to either two or three polynomials of degree at most t, respectively. These polynomials can then be used to reconstruct a larger degree polynomial $A(x)$ (of degree $2t$ or $3t$) as well as the check digit. Note that the definition of a check digit described here is slightly different to [67] in that for our schemes $l = S$. Thus we compare the reconstructed secret to the check digit, if the two are not equal then we reject S' and conclude that cheating has occurred. This is explained in greater detail in the following sections.

2.4.1 Proposed OKS-Secure Scheme

In this scheme a polynomial, $A(x)$ of degree at most $2t$, where $A(0) = S$, is used to construct two polynomials, $f(x)$ and $g(x)$ of degree at most t, which are then used to distribute shares to participants. Upon reconstruction both $f(x)$ and $g(x)$ are reconstructed and used to reconstruct $A(x)$. In order to check that no cheating has occurred, $\mathcal{C}$ checks that $A(0) = f(0) \cdot g(0)$. As with the original JS scheme, all computations are performed in $\mathbb{F}_q$ where q is a prime number and $q > 2n$. The `SHARE` and `REC` algorithms are as follows:

`SHARE`$(S) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $A(x) = S + Q_1 x + \cdots + Q_{2t} x^{2t}$ where $Q_j \in \mathbb{F}_q$ for $1 \leq j \leq 2t$. He then uses the first $2t + 1$ points on $A(x)$ to construct two more polynomials in the following fashion:

$$f(x) = A(1) + A(2)x + \cdots + A(t+1)x^t$$

$$g(x) = \alpha + A(t+2)x + \cdots + A(2t+1)x^t$$

Where α is picked such that $S = A(1) \cdot \alpha$ and the other coefficients of both $g(x)$ and $f(x)$ are points on $A(x)$. As a result of this we also require that

$A(1) \neq 0$ ($\mathcal{D}$ can simply re-sample $A(x)$ to achieve this).

Participant P_i is assigned the share $V_i = (f(i), g(i))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow S'$: A subset, γ, of $t+1$ participants (of which up to t can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$ and $g'(x)$ using Lagrange interpolation. $\mathcal{C}$ then uses the coefficients of $f'(x)$ and $g'(x)$ (specifically the points $A'(1), \cdots, A'(2t+1)$) to reconstruct $A'(x)$. If $A'(0) = f'(0) \cdot g'(0)$ then $\mathcal{C}$ outputs $A'(0)$ as the secret, otherwise output $\perp$.

Security Discussion

To prove the security of the proposed **OKS-secure scheme** we begin by showing that it is perfect and then prove that except with probability δ a set of dishonest participants cannot cheat the honest participants into accepting a false secret.

Theorem 1. *The proposed **OKS-secure scheme** is perfect, i.e., fewer than $t+1$ participants cannot reduce their uncertainty of the secret S.*

Proof. Say that the first t participants with shares $V_i = (f(i), g(i))$ for $1 \leq i \leq t$ wish to compute S. By pooling their shares and forming a coalition they can construct the following system:

$$
\begin{aligned}
f(1) &= A(1) + A(2) + \cdots + A(t+1) \\
g(1) &= \alpha + A(t+2) + \cdots + A(2t+1) \\
f(2) &= A(1) + 2 \cdot A(2) + \cdots + 2^t \cdot A(t+1) \\
g(2) &= \alpha + 2 \cdot A(t+2) + \cdots + 2^t \cdot A(2t+1) \\
&\vdots \\
f(t) &= A(1) + (t) \cdot A(2) + \cdots + (t)^t \cdot A(t+1) \\
g(t) &= \alpha + (t) \cdot A(t+2) + \cdots + (t)^t \cdot A(2t+1)
\end{aligned}
$$

For any value i it is known that $A(i) = S + i \cdot Q_1 + \cdots + i^{2t} \cdot Q_{2t}$. So the above system of equations can therefore be rewritten purely in terms of the coefficients of $A(x)$

(including S). This results in an unsolvable system composed of $2t$ independent equations and $2t+1$ unknowns. To clarify this, let $S = Q_0$ then, consequently, α becomes:

$$Q_0 \cdot (A(1))^{-1} = Q_0(\sum_{i=0}^{2t} Q_i)^{-1}$$

Therefore, the shares of a given participant in the coalition, denoted as P_w for $1 \leq w \leq t$, can be rewritten as:

$$f(w) = \sum_{j=1}^{t+1} \left(w^{j-1} \cdot \sum_{i=0}^{2t} j^i Q_i\right)$$

$$g(w) = Q_0(\sum_{z=i}^{2t} Q_i)^{-1} + \sum_{j=1}^{t+1} \left(w^{j-1} \cdot \sum_{i=0}^{2t} (j+t)^i Q_i\right)$$

The above can be represented as a system of equations featuring the unknowns of $Q_0, Q_1, \cdots, Q_{2t}$ (as the w values are known to the attackers). By the properties inherent in Shamir's secret sharing scheme; due to the uniform and independent choices of $Q_1, \cdots, Q_{2t}$ in $\mathbb{F}_q$ any $2t$ evaluations of $A(x)$ at non-zero distinct points are uniform and independently distributed in $\mathbb{F}_q$, for any choice of secret $S = A(0)$. In particular, the $2t$ evaluations $A(1), \cdots, A(t+1), A(t+2), \cdots, A(2t+1)$, are uniform and independent in $\mathbb{F}_q$, for any $S = A(0)$. Therefore the t non-constant coefficients of $f(x)$ and $g(x)$ respectively are uniform and independent in $\mathbb{F}_q$, hence by Shamir's scheme's properties the shares $f(i)$, and $g(i)$ for $i = 1, \cdots t$ are uniform for any choice of secrets $A(1)$ and α shared by $f(x)$ and $g(x)$ respectively, and hence also for any choice of the secret S.

Put simply, each of the t participants in the coalition is essentially given 2 shares of a degree $2t$ polynomial, albeit in roundabout fashion. Therefore, the coalition has a set of $2t$ shares relating to a polynomial of degree $2t$. Due to the perfectness of Shamir's secret sharing scheme [97], they cannot compute any information relating to S as (from the point of view of the t participants) all potential q amount of values of S are equally likely. □

Theorem 2. *The proposed* ***OKS-secure scheme*** *is a* (t, n, δ) *SSCD under the OKS model with share size* $|V_i| = 2\log(q)$*, secret domain* $\mathcal{S} = \mathbb{F}_q$ *and probability of cheating* $\delta = \frac{1}{q}$*.*

Proof. Without loss of generality assume that it is the first t participants $(P_1, \cdots, P_t)$ who wish to cheat. Here, by cheating, we mean that the set of (at most t) participants submit modified shares that pass the test and from the resulting value (S') released by the combiner, the cheating participants learn the actual secret (S), while the honest participants learn nothing.

In order to successfully cheat, this set of t dishonest participants, denoted by θ, will need to not only force the reconstruction of a secret $S' = S + \beta_1$ but also ensure that $S' = f'(0) \cdot g'(0)$ where $f'(0) = f(0) + \beta_2$ and $g'(0) = g(0) + \beta_3$ such that β_1 , β_2 , and β_3 are some constants. In order to cheat they need to pick these values such that:

$$S + \beta_1 = (f(0) + \beta_2)(g(0) + \beta_3)$$

Whilst it is trivial for θ to use the Tompa and Woll attack force the reconstruction of $f'(0)$ and $g'(0)$ with any values they choose for (respectively) β_2 or β_3, we note that this is not at all the case for S'. That is, ignoring the difficulty that θ face in modifying their shares in such a way that coefficients of the resulting polynomials $f'(x)$ and $g'(x)$ are points of a $2t$-degree polynomial $A'(x)$ with a secret $S' = A'(1) \cdot \alpha'$ (i.e., a $2t$-degree polynomial that satisfies $2t + 2$ points); learning S implies that θ know the value of β_1. To explain this we can expand the equation above to get:

$$\beta_1 = A(1) \cdot \beta_3 + \beta_2 \cdot g(0) + \beta_2 \cdot \beta_3$$

.

This can be simplified further by noting that the cheaters really only need to change one of either $f(x)$ or $g(x)$. Therefore, without loss of generality we can set

$\beta_3 = 0$ and simplify the equation to

$$\beta_1 = \alpha \cdot \beta_2$$

. Summarising the above means that to successfully cheat the coalition will have to:

1. Compute a $2t$-degree polynomial $A'(x)$ from the coefficients of $g(x)$ and $f'(x)$ such that $S' = g(0) \cdot f'(0)$; to restate, they must compute said $2t$-degree polynomial from $2t + 2$ points.

2. Even if $f'(x)$ and $g(x)$ achieve the above condition the cheaters must also compute $f'(x)$ such that $S' = S + \alpha \cdot \beta_2$ which is an equation with 2 unknowns (S and α).

Discounting the difficulty in computing $f'(x)$ to result in the required $2t$-degree polynomial, the cheater's chances of cheating rely on reducing the unknowns of the equation specified in condition 2 above. Since, by the proof of theorem 1, they cannot do this, their chances of cheating are analogous to guessing, i.e., $\frac{1}{q}$. Additionally, each participant gets, as their share, two field elements from a field of characteristic q, thus the share size is $2 \log q$. □

2.4.2 Proposed CDV-Secure Scheme

In our **OKS-secure scheme** it is easy to see that if the cheaters know the value of S then it is entirely possible for them to compute $\beta_1 = \beta_3 \cdot f(0) + \beta_2 \cdot g(0) + \beta_2 \cdot \beta_3$ and therefore break the security of the system. In fact if the cheaters know S then they can actually compute the share of the honest participant, as they will be able to solve the system given in the proof of Theorem 1.

So, while this scheme is certainly secure in the OKS setting (where the cheaters do not know the secret) it is not at all secure in the CDV setting (where they do know the secret). However, it is possible to rectify this with a simple modification.

The construction and cheating detection mechanism for the **CDV-secure scheme** is much the same as for the **OKS-secure scheme**. However, here $A(x)$ is of degree at most $3t$ and we use the first $3t+1$ points on $A(x)$ to construct three polynomials $f(x)$, $g(x)$ and $h(x)$ of degree at most t. This means that each participant obtains three shares, one relating to each of the degree t polynomials. As per the second scheme described in [71] this prevents malicious participants (who know the secret) from computing the honest participant's share. We define the check digit as $l = f(0) \cdot g(0) \cdot h(0)$. All computations are performed in $\mathbb{F}_q$ where q is a prime number such that $q > 3n$ and $|\mathcal{S}| \leq q$.

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $A(x) = S + Q_1 x + \cdots + Q_{3t} x^{3t}$. He then uses the first $3t+1$ points on $A(x)$ to construct three more polynomials:

$$f(x) = A(1) + A(2)x + \cdots + A(t+1)x^t$$

$$g(x) = \alpha + A(t+2)x + \cdots + A(2t+1)x^t$$

$$h(x) = \omega + A(2t+2)x + \cdots + A(3t+1)x^t$$

Where α and ω are picked such that $S = A(1) \cdot \alpha \cdot \omega$ and the other coefficients of $f(x)$, $g(x)$ and $h(x)$ are all points on $A(x)$. Participant P_i is assigned the share $V_i = (f(i), g(i), h(i))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow S'$: A subset, γ, of $t+1$ participants (of which up to t can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$, $g'(x)$ and $h'(x)$ using Lagrange interpolation. $\mathcal{C}$ then uses the coefficients of $f'(x)$, $g'(x)$ and $h'(x)$ (specifically the points $A'(1), \cdots, A'(3t+1)$) to reconstruct $A'(x)$, again using Lagrange interpolation. If $A'(0) = f'(0) \cdot g'(0) \cdot h'(0)$ then $\mathcal{C}$ outputs $A'(0)$ as the secret, otherwise output $\perp$.

Security Discussion

As before, in this section the security of the given scheme is proved.

Theorem 3. *The proposed* ***CDV-secure scheme*** *is perfectly secure.*

Proof. Analogous to the proof of Theorem 1. □

Theorem 4. *The proposed* ***CDV-secure scheme*** *is a* (t, n, δ) *SSCD under the CDV model with share size* $|V_i| = 3\log(q)$*, secret domain size* $|\mathcal{S}| \leq q$ *and probability of cheating* $\delta = \frac{1}{q}$*.*

Proof. As with the proof for Theorem 2 assume that we have $t+1$ participants who wish to reconstruct S and that t of these participants make up the set of dishonest participants denoted by θ. All computations are done in the field $\mathbb{F}_q$.

In order to successfully cheat, θ need to force reconstruction of $S' = f'(0) \cdot g'(0) \cdot h'(0)$. Where we define $S' = S + \beta_1$, $f'(0) = f(0) + \beta_2$, $g'(0) = g(0) + \beta_3$ and $h'(0) = h(0) + \beta_4$. Since θ know the value of S they can easily compute S'. So to successfully cheat they must solve the following equation:

$$S' = (f(0) + \beta_2)(g(0) + \beta_3)(h(0) + \beta_4)$$

In order to solve this equation they must first compute $f(0)$, $g(0)$ and $h(0)$. We show that this is not possible except with probability $\frac{1}{q}$ which is analogous to guessing. Without loss of generality assume that θ is composed of participants $P_1, \cdots, P_t$ and P_{t+1} is the honest participant. By pooling their shares, θ can construct the following

system:

$$f(1) = A(1) + A(2) + \cdots + A(t+1)$$

$$g(1) = \alpha + A(t+2) + \cdots + A(2t+1)$$

$$h(1) = \omega + A(2t+2) + \cdots + A(3t+1)$$

$$f(2) = A(1) + 2 \cdot A(2) + \cdots + 2^t \cdot A(t+1)$$

$$g(2) = \alpha + 2 \cdot A(t+2) + \cdots + 2^t \cdot A(2t+1)$$

$$h(2) = \omega + 2 \cdot A(2t+2) + \cdots + 2^t \cdot A(3t+1)$$

$$\vdots$$

$$f(t) = A(1) + (t) \cdot A(2) + \cdots + (t)^t \cdot A(t+1)$$

$$g(t) = \alpha + (t) \cdot A(t+2) + \cdots + (t-1)^{t-1} \cdot A(2t+1)$$

$$h(t) = \omega + (t) \cdot A(2t+2) + \cdots + (t)^t \cdot A(3t+1)$$

As with the proof of theorem 1, the above system can be represented in terms of the coefficients of $A(x) = S + Q_1 x + \cdots + Q_{3t} x^{3t}$ and ω (as $\alpha = \frac{S}{\omega \cdot A(1)}$). Since S is known, the system is composed of $3t$ equations and $3t+1$ unknowns, meaning that θ cannot compute the values of $f(0)$, $g(0)$ or $h(0)$ except with probability $\frac{1}{q}$, i.e., they attempt to guess a value for either of the three variables. Therefore we can define the probability of successful cheating as $\delta = \frac{1}{q}$.

Since each participant receives three values as their share and all values (except for $S \in \mathcal{S}$) are drawn from $\mathbb{F}_q$ then the size of the share size is $|V_i| = 3\log(q)$. Thus the proposed scheme is a (t, n, δ) SSCD under the CDV model. □

2.5 Conclusion and Comparison

The results presented in this Chapter are threefold:

1. It was shown that the two SSCD schemes presented in [71] are vulnerable to the classic attack described by Tompa and Woll [99].

2. The construction of a near optimal SSCD scheme secure under the OKS model

that supports arbitrary finite fields and a small secret domain was given. This is the most efficient of such schemes to date, being the same size as the original scheme devised in [71], which they state is 2 bits greater than optimal. Thus it is 1 bit more efficient than the scheme given in [86] which is the only other secure scheme that achieves these extended capabilities.

3. A near optimal SSCD scheme secure in the CDV model that has the same share size as the insecure scheme described in [71] was given. This is an extremely efficient scheme as it achieves a share size equal to that of the scheme presented by Cabello et al. [17] which is the most efficient yet.

We note that although our schemes achieve a small share size, unfortunately, our OKS scheme requires double the amount of field multiplications during reconstruction than the OKS scheme devised in [86] (due to the extra reconstruction of a $2t$ degree polynomial). Likewise our CDV scheme requires double the amount of field multiplications for reconstruction than the CDV scheme given by [17]. An open problem within this field is the construction of a SSCD scheme secure under the CDV model that achieves optimum share size. Another interesting topic recently introduced in [86] is the construction of a SSCD scheme secure in the OKS model that is not only optimal but also supports an arbitrary field and a small secret domain. In the next Chapter we take an alternate look at the construction and specification of SSCD schemes, particularly focusing on the role of the combiner and the security assumptions implicit in such an assumption.

Chapter 3

Outsourced Cheating Detection for Secret Sharing

3.1 Introduction

Within the literature of SSCD the focus has been on producing ever more efficient schemes. However, we note that a common feature across all SSCD protocols, that has yet to be addressed, is that even if cheating is detected, the cheaters still get the secret. Specifically, if an invalidated secret, S' is reconstructed then the cheating participants (who caused reconstruction of the invalid secret) will still be able to obtain the original secret (S), even if their cheating has been successfully detected.

In this Chapter, we propose a solution designed to overcome this problem: outsourced SSCD (OSSCD). Our proposed protocol utilises the same techniques as SSCD, however, before the secret is reconstructed we have participants distribute their shares among a set of special validation servers. These validation servers then perform a public computation to determine if cheating has occurred. They do this without reconstructing S. The result of this is that, if cheating has occurred, the servers can halt the protocol, ensuring no one learns the secret.

We present two efficient constructions of our proposed OSSCD protocol, one capable of detecting cheating with high probability, the other capable of tolerating many secrets simultaneously. As before, we begin by defining a simple threshold secret sharing scheme.

Definition 1. *A (t, n) threshold secret sharing scheme consists of n participants, $P_1, \cdots, P_n$ who are privately assigned shares of D's secret value S, which is across some finite field $\mathbb{F}$. Correctness is achieved if any set of $t+1$ or more participants*

can recover S. Security is achieved if a set of t or less participants cannot reduce their uncertainty of S.

This then naturally leads back to the division of a secret sharing scheme into two algorithms, `SHARE` and `REC`, where D executes `SHARE` and a third party, the combiner, C, executes `REC` (as defined in the previous Chapter).

If $|\gamma| \geq t + 1$ and all participants follow the protocol exactly, then $S' = S$, otherwise the probability of computing the correct secret (with less than t participants) is analogous to guessing, i.e., $1/|\mathbb{F}|$. This provides a high level of privacy for D. We note, however, that such schemes are dependent on the assumption that all participants follow the protocol exactly. As we know, this is, of course, not always the case and to counter this SSCD was proposed and researched extensively [1, 17, 20, 29, 67, 86, 89], with most work focusing on producing constructions with low share size (the maximum amount of information each participant must store).

Although the field of SSCD is well established, there are many conflicting implementations and security frameworks. Thus, in 2008 Cramer et al. [40] generalised SSCD by proposing, what they call, "algebraic manipulation detection codes" (AMD codes). They showed that SSCD schemes can be constructed by applying an AMD code to a secret sharing scheme (similar, but far more rigorous, to the notion of check digit functions presented in the preceding Chapter). Their approach formalises many of the notions present within the literature of SSCD and allows for a more formal and easier analysis of results and security. In general, however, detecting cheating via SSCD or AMD codes is performed in exactly the same fashion. That is, participants are given some extra information, along with their shares, which enable them to check the validity of a reconstructed secret during the execution of the `REC` algorithm.

The problem with this approach is that it still allows the cheaters to learn the se-

cret. Therefore, to combat this, we propose a new type of cheater detection protocol, in which the function of the combiner does not reside with just one individual. Our proposed protocol makes use of multiple third parties who will check the validity of a given secret without ever actually reconstructing the secret itself. We call such a protocol an outsourced SSCD (OSSCD) scheme. In the next sections we review SSCD and AMD codes, following this, we discuss our contribution in depth.

3.1.1 Background

For completeness we reiterate what was examined in the previous Chapter, defining SSCD in general, as well as threshold SSCD protocols.

SSCD

A (t, n, ϵ) SSCD scheme can be defined as a (t, n) threshold secret sharing scheme with a modified `REC` algorithm.

`SHARE`$(S) \rightarrow (V_1, \ldots, V_n)$:

A probabilistic algorithm wherein D takes a secret S and computes n shares: $V_1, \ldots, V_n$. Participant P_i is privately assigned the share V_i, for $1 \leq i \leq n$.

`REC`$(\gamma) \rightarrow (S')$ **or** $(\perp)$:

A set of exactly $t+1$ participants, γ, submit their shares to C who performs a computation in order to determine if any modified shares have been submitted. If cheating is detected then C halts the protocol and outputs a special symbol, $\perp$. Otherwise C computes and outputs a secret, S'.

Note that we use S' to show that the constructed secret may or may not be equal to S.

Definition 2. *A (t, n, ϵ) SSCD scheme is a (t, n) secret sharing scheme secure against a set of at most t cheaters. If no cheating occurs then $Pr[S = S'] = 1$.*

Otherwise, in the case of cheating $Pr[\texttt{REC}(\gamma) = \perp] \geq 1 - \epsilon$*, conversely* $Pr[\texttt{REC}(\gamma) = S' \mid S \neq S'] \leq \epsilon$.

We note that, unlike D, the combiner, C, is not intrinsically trusted by the participants. Rather, in most SSCD protocols it is assumed that all data submitted to C and all computations performed by C are public knowledge. As a result of this it is easy to see that, although cheating can be detected with high probability, the cheaters will, unfortunately, learn the secret. To clarify this, say a set of cheaters force reconstruction of a secret $S' = S + \Delta_s$. Even if they are detected in their cheating the value of S' is publicly revealed, which subsequently allows the cheaters to learn S as it is a trivial matter for the cheaters to control the exact value of Δ_s (as per Tompa and Woll [99]).

If we consider the original use case that Shamir proposed for secret sharing, i.e., protecting cryptographic keys at rest, we can easily see how disastrous allowing a set of cheaters to learn the secret before all other (honest) participants is. If said cryptographic key was used to encrypt private or sensitive information then the cheaters of the secret sharing scheme would have instant and unrestricted access to all of the information, whilst the honest parties would be at a distinct disadvantage, having no access at all.

Furthermore, if instead participants privately submitted their shares to C then this would allow C to simply reconstruct the secret himself, without necessarily revealing it to the participants. This forces the participants to place a high degree of trust in C which is not an ideal situation.

AMD Codes

AMD codes can be viewed as an abstraction of older techniques utilised within the literature for the purpose of detecting tampering or cheating in secret sharing

and multi-party computation. These codes were originally introduced by Cramer et al. in [39]. An AMD code essentially allows for the detection of algebraic tampering on a stored value. To explain this we can use the description from [41].

Consider an abstract storage device $\Sigma(\mathcal{G})$ that stores a value X, such that $X \in \mathcal{G}$ where $\mathcal{G}$ is publicly known an abelian group. A given adversary is able to algebraically manipulate this stored value by adding some chosen value $\Delta \in \mathcal{G}$ such that $\Sigma(\mathcal{G})$ now holds $X + \Delta$. An AMD code consists of a source, S which is encoded as X and stored in $\Sigma(\mathcal{G})$ in such a way that the adversary's manipulation can be detected with high probability. We can define these codes using the definition given in [41]:

Definition 3. *[41] Let $\mathcal{S}$ be a set of size $|\mathcal{S}| > 1$ and $\mathcal{G}$ an abelian group of order $|\mathcal{G}|$. Consider a pair (E, D) where E denotes a probabilistic encoding map $E : \mathcal{S} \to \mathcal{G}$ and D denotes a deterministic decoding map $D : \mathcal{G} \to S \cup \{\perp\}$ such that $D(E(S)) = S$ with probability 1 for every $S \in \mathcal{S}$.*

The pair (E, D) is an $(|\mathcal{S}|, |\mathcal{G}|, \epsilon)$-AMD code if, for every $\Delta \in \mathcal{G}$ and every $S \in \mathcal{S}$ it holds that:

$$Pr[(D(E(S) + \Delta) \notin \{S, \perp\}] \leq \epsilon$$

A systematic AMD code [41] occurs when $\mathcal{S}$ is a group, $\mathcal{G} = \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2$ where $\mathcal{G}_1$ and $\mathcal{G}_2$ are groups and E, the encoding mapping, is of the form:

$$E(S) = (S, X, f(X, S))$$

Where $X \in \mathcal{G}_1$ is chosen uniformly at random and the function f is given as $f : \mathcal{G}_1 \times \mathcal{S} \to \mathcal{G}_2$. From this we define the decoding algorithm as:

$$D(S, X, e) = \begin{cases} S, \text{ if } e = f(X, S) \\ \perp, \text{ otherwise.} \end{cases}$$

It is easy to see that the above explanation of a systematic AMD code is equivalent to an SSCD scheme. In fact a given (t, n, ϵ) SSCD scheme, with secret domain $|\mathcal{S}|$ and share size $|\mathcal{G}|$, can be constructed by utilising a systematic $(|\mathcal{S}|, |\mathcal{G}|, \epsilon)$-AMD code. We can achieve this by simply running the encoding algorithm, E (the function $f(X, S)$) on a secret $S \in \mathcal{S}$ and a given value $X \in \mathcal{G}_1$ to get a value $y \in \mathcal{G}_2$. Each participant is then given a share of S, X and y. To detect cheating we simply reconstruct these values and then run the decoding algorithm.

In their seminal work Cramer et al. [39] introduced AMD codes as both a generalisation and formalisation of many different algebraic techniques used within the literature of secret sharing for the purpose of both cheater detection and identification. AMD codes are a more natural approach to this field of work and are simpler to both demonstrate and prove the security of. As such, we make extensive use of the notation and concepts found within the AMD code literature. We note that AMD codes do not replace SSCD, rather they allow for a more natural description and construction of such protocols.

3.1.2 Our Contribution

Our proposed OSSCD protocol utilises the same `SHARE` protocol as regular SSCD, however we make use of a modified `REC` algorithm. To clarify, OSSCD replaces the combiner, C, with a set of special validation servers. Instead of having participants simply send their shares to C we instead have them utilise a re-sharing protocol to distribute their shares to a set of servers. The validation servers then collectively run a validation algorithm that determines whether or not cheating has occurred. This is done without reconstructing the secret. If cheating has occurred then the protocol halts and no one learns the secret. Otherwise, the servers allow the participants to learn the secret.

To achieve this we utilise a slightly modified validation technique commonly

used in MPC protocols [47] (MACs) as well as the re-sharing technique given in [52]. We demonstrate two constructions of our proposed OSSCD protocol that are based on Shamir's secret sharing scheme [97] and strong, systematic AMD codes for the purpose of detecting cheating.

The reasoning behind this is that during the validation phase of our protocol the servers must publicly reconstruct some information. Specifically, they reconstruct X in a systematic AMD code. If we were to utilise a normal AMD code, then reconstructing X would allow the cheaters to learn the secret.

Our new construction is in the same vein of thought as recent works such as [44, 98] which utilise a set of special, independent (i.e., non- maliciously collaborating) servers to perform critical tasks for a set of participants. This is a realistic and real-world assumption that is especially relevant in light of the infrastructure and model of the internet. Wherein a set of participants can purchase the service of a set of chosen servers and be relatively certain of the non-malicious nature of said servers (i.e., the servers will not deliberately supply false information).

A similar method that could be used to achieve the same result is to have participants enact a multi-party computation amongst themselves. However, such protocols would require far more information be given to each participant, furthermore the communication costs among the participants would also increase.

Our solution allows for low complexity and communication among participants and does not require a full blow multi-party computation protocol with a set of private and authenticated communication channels between each participant (we only require secure communication between the servers and the participants). This is particularly desirable when participants have relatively low computational power compared to servers, a scenario that can be likened to, for instance, clients with mobile phones (the participants) offloading the computation to the cloud (the servers).

3.1.3 Outline

The rest of this Chapter is organised as follows. In the next section (section 3.2) we define a model for our new protocol. Following this, section 3.3 displays some of the tools needed in our protocol. In section 3.4 we show a specific construction of our protocol that is capable of detecting cheating, with a high probability, for a single secret. Lastly, section 3.5 extends on this, giving a construction that can be used to verify multiple different secrets.

3.2 Model

Our OSSCD protocol consists of a dealer, D, a set of n participants, $P_1, \cdots, P_n$ and m validation servers, $A_1, \cdots, A_m$. The basic premise of OSSCD is simple. As with SSCD, participants are assigned shares by D. However, to reconstruct the secret participants must first redistribute their shares to a set of $k+1$ out of the m servers ($k < m$). These servers then collectively run a validation algorithm to determine if cheating has occurred. If cheating has occurred then the protocol is halted and no one learns the secret, otherwise the servers publicly reconstruct and broadcast the secret. The proposed protocol can be divided into three algorithms or phases:

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$:

A probabilistic algorithm wherein D takes a secret S, as well as some validation information (specifically, X and y in an AMD code), and computes n shares: $V_1, \ldots, V_n$. Participant P_i is privately assigned the share V_i, for $1 \leq i \leq n$.

$\texttt{RESHARE}(\gamma) \rightarrow (\{Y_j\}_{A_j \in \omega})$:

A probabilistic algorithm in which a set of exactly $t+1 \leq n$ participants, γ, privately distribute their shares to a set of servers, denoted as ω, of size $k+1$ (out of m, where $k < m$). Specifically, $P_i \in \gamma$ assigns to server $A_j \in \omega$ the

share V_{i_j} of his original share V_i. Each server then performs a computation to determine his share, denoted as Y_j.

$\texttt{VAL} \rightarrow (S'), (\perp)$:

A deterministic algorithm in which the set of servers, ω, perform a joint computation across their shares, Y_j, in order to detect if cheating has occurred. If cheating has occurred the servers output $\perp$, otherwise they publicly reconstruct S', which is taken to be the secret.

The essential premise of the validation algorithm is that we have servers compute a random number, R as well as reconstructing the value, X (from the AMD code). The servers then perform basic MPC techniques across this public information and their private share, to securely compute the value:

$$R \cdot y - R \cdot f(S, X)$$

If the reconstructed value is not 0 then it is assumed that cheating has occurred. This technique is a simplified version of the "MACCheck" protocol found in [47].

Similar to the SSCD assumption regarding C, as well as the model depicted in [44, 98], we assume that the validation servers follow the protocol exactly. However, we also assume that subsets of k or less of these servers may try to learn some extra information by pooling their resources (i.e., they are semi-honest). This is encapsulated in the following definition:

Definition 4. *A (t, n, k, ϵ) OSSCD scheme can be defined as a (t, n, ϵ) SSCD scheme in which a set of $k+1$ out of m servers carry out the functions of the combiner. The scheme is secure if a set of t participants and a (separate) set of k servers cannot reduce their uncertainty of S. If cheating is detected then neither the participants nor the servers should learn the secret, otherwise the secret is broadcast by the servers.*

As previously stated, we do not assume that either the participants or the servers know S. In terms of communication, we assume there exists secure channels between each server and participant. However, no such channels exist between each of the participants. Rather, we assume that the `VAL` algorithm is carried out by the servers over a public broadcast channel, visible to both participants and servers.

3.3 Preliminaries

In this section we go through some preliminaries necessary for the OSSCD constructions given in sections 3.4 and 3.5.

3.3.1 AMD Code Constructions

To detect cheating in our two constructions we utilise two different AMD codes. The first of these was originally given by Cabello et al. [18], in the form of an SSCD scheme. Cramer et al. [42] later described this protocol in terms of AMD codes, dubbing it the "multiplication AMD code." As before, let the encoding function of a systematic AMD code be represented by the function $f(X, S)$, the multiplication AMD code simply takes the form

$$f(X, S) = S \cdot X$$

If we set $S, X \in \mathbb{F}_q$, where q is a prime number, then this gives a $(q, q^2, 1/q)$ systematic AMD code.

The second code we utilise is a variation of the code that was originally given by Cramer et al. [39] in their seminal work on AMD codes. We utilise this code for the verification of multiple secrets, $S_1, \cdots, S_w \in F_q$, where $q > w$ and is a prime number, as before. Let $X \in \mathbb{F}_q$, then the encoding function for this code is as follows:

$$f(X, S) = X^{w+2} \sum_{l=1}^{w} S_l \cdot X^l$$

This gives a $(q^w, q^{w+2}, (w+1)/q)$ systematic AMD code.

3.3.2 Dynamic Re-sharing Protocol

In order to carry out the `RESHARE` algorithm we utilise the resharing protocol originally detailed by Desmedt and Jajodia [52]. Their scheme allows for the conversion of a (t, n) threshold secret sharing scheme to a (k, m) threshold secret sharing scheme, where k and m can be greater than, less than or even equal to t and n respectively. Assume that we have a set of participants, $P_1, \cdots, P_n$ in a (t, n) Shamir secret sharing scheme, with respective shares $V_1, \cdots, V_n$ relating to $f(x)$ with $f(0) = S$. Denote a set, of size $t+1$, of these participants as γ. Now, say that the members of γ wish to redistribute their shares to a different set of m participants, $A_1, \cdots, A_m$ in the form of a (k, m) scheme. To do this γ would need to carry out the following protocol with a set of $k+1$ out of the m other participants, denoted as ω:

1. $P_i \in \gamma$ computes:
$$B_i = V_i \Bigg(\prod_{\substack{P_i, P_l \in \gamma \\ i \neq l}} \frac{-l}{i-l} \Bigg)$$

2. Each P_i then privately distributes shares of B_i to all $A_j \in \omega$ using a polynomial of degree at most k (as per Shamir's secret sharing scheme), such that A_j gets the share V_{i_j} from P_i.

3. A_j computes his share of S as:
$$Y_j = \sum_{P_i \in \gamma} V_{i_j}$$

3.3.3 Random Value Generation

In order for our scheme to work each of the servers must have access to a random value R, such that this value is known and privately held by all servers. There are

many such methods that could be employed to compute this value securely, for instance each server picks a random value r and then securely distributes this value to all of the other servers. The value R is then the summation of all r values.

We note that, the servers could easily compute many such values ahead of time (allowing them to perform many OSSCD evaluations). As such, we will not go into detail here, but merely assume that each of the servers knows the random and arbitrary value R.

3.4 OSSCD Scheme Based on The Multiplication AMD Code

In this section we introduce an OSSCD construction based on the multiplication AMD code [18] introduced in section 3.3.1. As before, we have a set of n participants, $P_1, \cdots, P_n$, a dealer, D, and m servers, $A_1, \cdots, A_m$. All computations are performed in the finite field $\mathbb{F}_q$ where q is a prime number such that $q > \max(n, m)$.

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$:

D takes a random number, X, such that $X \neq 0$ and computes $y = X \cdot S$. He then computes three random polynomials, $f(x)$, $g(x)$ and $h(x)$, of degree at most t where $f(0) = S$, $g(0) = X$ and $h(0) = y$. Using Shamir's secret sharing scheme he distributes shares of these polynomials to each of the n participants, with P_i receiving $V_i = (f(i), g(i), h(i))$ for $i = 1, \cdots, n$.

$\texttt{RESHARE}(\gamma) \rightarrow (\{Y_j\}_{A_j \in \omega})$:

A set of exactly $t + 1$ participants, denoted as γ, conduct the re-sharing protocol described in section 3.3.2 with a set of $k + 1$ servers, denoted as ω. A given server, $A_j \in \omega$ is assigned the share V_{i_j} by participant P_i, where $V_{i_j} = (f_i(j), g_i(j), h_i(j))$ and $f_i(x)$, $g_i(x)$ and $h_i(x)$ are the random, degree k polynomials computed by a given P_i. Each A_j then computes his share as: $Y_j = (F(j), G(j), H(j))$ where $F(0) = S$, $G(0) = X$ and $H(0) = y$. As per the

re-sharing protocol:

$$F(x) = \sum_{P_i \in \gamma} f_i(x)$$
$$G(x) = \sum_{P_i \in \gamma} g_i(x)$$
$$H(x) = \sum_{P_i \in \gamma} h_i(x)$$

For clarity we can denote A_j's share as:

$$Y_j = (S_j, X_j, y_j)$$

$\texttt{VAL} \rightarrow (S')$ **or** $(\perp)$:

Each $A_j \in \omega$ publicly broadcasts X_j (their share of $G(x)$). Using these shares they compute $X' = G'(0)$. If $X' = 0$ they halt the protocol and output $\perp$. Otherwise each A_j then collectively computes: $\alpha = R \cdot X'$, where R is the random value mentioned in section 3.3.3.

Following this each A_j then privately computes a value $\beta_j = R \cdot y_j$. He then broadcasts the value $\upsilon_j = \beta_j - S_j \cdot \alpha$. Collectively, ω then reconstructs the value υ from the shares υ_j. If $\upsilon \neq 0$ the protocol halts and the output is $\perp$, otherwise the servers reconstruct and broadcast the value S' from their shares S_j.

Theorem 5. *The proposed construction is a secure* (t, n, k, ϵ) *OSSCD scheme with error probability* $\epsilon = \frac{1}{q-1}$.

We know that the scheme is secure against t participants during the sharing phase due to the proof in [18]. Due to the proof in [52], we also know that the re-sharing protocol is secure. It remains to be seen, however, if the scheme is secure against t participants and k servers during the validation phases.

Proof. We will first prove that the scheme is secure against a set of t participants and a separate set of k servers when no cheating has occurred. Without loss of generality,

assume that the set γ is made up of the first $t+1$ participants, $P_1, \cdots, P_{t+1}$ and that it is the first t of these that forms a coalition. After the verification phase has been conducted, but before reconstruction, the set of t participants will have the following information (in addition to their shares of X, y and S):

1. The re-shared values each of the participants privately sent to the servers, of the form:

   $$V_{i_j} = (f_i(j), g_i(j), h_i(j))$$

   for $A_j \in \omega$ and $i = 1, \cdots, t$.

2. The publicly reconstructed X', here $X' = X$ as we assume no cheating has occurred.

3. k public shares of the form:

   $$\upsilon_j = \beta_j - S \cdot \alpha$$

   which can be rewritten as:

   $$\upsilon_j = R(y_j - S \cdot X)$$

We can rewrite each y_j and S_j as:

$$y_j = \sum_{i=1}^{t+1} h_i(j) \quad S_j = \sum_{i=1}^{t+1} f_i(j)$$

Which gives us:

$$H(x) = \sum_{i=1}^{t+1} h_i(x) \quad F(x) = \sum_{i=1}^{t+1} f_i(x)$$

Each of the validation shares, υ_j, broadcast by the servers now takes the form:

$$\upsilon_j = R\left(\sum_{i=1}^{t+1} h_i(j) - \sum_{i=1}^{t+1} f_i(j) \cdot X \right)$$

Now, the coalition know the values of the first t of the shares these values are composed of, i.e., $h_1(j), \cdots, h_t(j)$ and the shares $f_1(j), \cdots, f_t(j)$. So let us rewrite the above equation by substituting the coalitions known values as:

$$\rho = \sum_{i=1}^{t} h_i(x) - f_i(j) \cdot X$$

We also make the following additional substitution:

$$\psi = R(f_{t+1}(j) - h_{t+1}(j) \cdot X)$$

Rewriting our equation as:

$$\upsilon_j = R \cdot \rho + \psi$$

It is easy to see that the above equation is unsolvable to the coalition without knowing either the values of R or ψ. They cannot compute R as no server will communicate with a participant outside the bounds of the protocol. The value ψ is also unknowable due to the very same reason (as $\psi = R(f_{t+1}(j) - h_{t+1}(j) \cdot X)$). Furthermore, even if the coalition did compute the value of ψ and were able to then compute a polynomial $T(x) = h_{t+1}(x) - f_{t+1}(x) \cdot X$, they would still not have enough information to compute S.

On that line of thought, we note that if the coalition were able to explicitly compute $h_{t+1}(x) - f_{t+1}(x)$ then this would allow them to compute a share to a polynomial with free term $y - S$, which in turn would allow them to compute S (as they know X). However this is not possible, since they do not know the values of $h_{t+1}(x)$ or $f_{t+1}(x)$ and the coefficients of these polynomials are essentially random, meaning that all possible values are equally likely.

As a result of this no information relating to S is gained by the coalition, as the free term of $T(x)$ is simply a share of a polynomial with free term equal to 0 (i.e., the same value as the verification value, υ). This is also the case if unsuccessful cheating occurs, however the value of υ will instead be equal to Δ i.e., the error introduced by the cheaters.

We note that the above section of the proof also holds for any set of k servers, as they hold even less information than the participants. In fact the servers only, collectively, hold two sets of k shares relating to two different k degree polynomials, meaning that they cannot compute any information relating to S. Therefore neither a set of t participants, nor a set of k servers can gain any information relating to S. It only remains to prove that the probability of cheating is $\epsilon = \frac{1}{q-1}$.

In order to cheat, the coalition will attempt to force the reconstruction of $S' = S + \Delta_s$ where $\Delta_s \neq 0$. To do this they will also need to force reconstruction of $y' = \Delta_y$ and $X' = X + \Delta_X$ such that:

$$y' = S' \cdot X'$$

Put simply, in order to cheat they will need to solve:

$$y + \Delta_y = R\Big((S + \Delta_s)(X + \Delta_X)\Big)$$

Which simplifies to:

$$\Delta_y = R(S\Delta_X + \Delta_s X + \Delta_s \Delta_X + SX) - y$$

Since the coalition do not know the value of X ahead of time they cannot solve the above equation and cannot compute appropriate values that will enable undetected cheating. Furthermore, the value R is unknown throughout the protocol, resulting in the coalition also being unable to compute S. In fact the probability of successful cheating is analogous to essentially guessing the value of X i.e., $1/q$. However, in our construction $X \neq 0$, meaning that they have a $1/(q-1)$ chance of guessing X. This, therefore, results in an error probability of $\epsilon = \frac{1}{q-1}$. □

3.5 Multi-Secret OSSCD Scheme

In this section we utilise the second of the AMD codes discussed in section 3.3.1 to construct an OSSCD scheme capable of tolerating multiple secrets at a time. We

take the same assumptions as the previous section, however, this time we have W secrets, $S_1, \cdots, S_W$ and all computations are performed in the finite field $\mathbb{F}_q$ where q is a prime number such that $q > \max(n, m, W)$.

$\texttt{SHARE}(S) \rightarrow (V_1, \ldots, V_n)$:

D takes a random number, X, such that $X \neq 0$ and computes

$$y = X^{W+2} + \sum_{l=1}^{W} S_l \cdot X^l$$

He then computes $W+2$ random polynomials, $g(x)$, $h(x)$ and $f_1(x), \cdots, f_W(x)$ of degree at most t where $f_l(0) = S_l$ for $l = 1, \cdots, W$, $g(0) = X$ and $h(0) = y$. Using Shamir's secret sharing scheme D distributes shares of these polynomials to each of the n participants, with P_i receiving:

$$V_i = (f_1(i), \cdots, f_W(i), g(i), h(i))$$

for $i = 1, \cdots, n$.

$\texttt{RESHARE}(\gamma) \rightarrow (\{Y_j\}_{A_j \in \omega})$:

A set of exactly $t+1$ participants, denoted as γ, conduct the re-sharing protocol described in section 3.3.2 with a set of $k+1$ servers, denoted as ω. A given server, $A_j \in \omega$ is assigned the share V_{i_j} by participant P_i, where $V_{i_j} = (f_{1_i}(j), \cdots, f_{W_i}(j), g_i(j), h_i(j))$ and $f_{1_i}(x), \cdots, f_{W_i}(x)$, $g_i(x)$ and $h_i(x)$ are the random, degree k polynomials computed by a given P_i. Each A_j then uses the latter half of the re-sharing protocol to compute his share as:

$$Y_j = (F_l(j), G(j), H(j))$$

where $l = 1, \cdots, W$, $F_l(0) = S_l$, $G(0) = X$ and $H(0) = y$. As before, for clarity we can denote A_j's share as $Y_j = (S_{1_j}, \cdots, S_{W_j}, x_j, y_j)$.

$\texttt{VAL} \rightarrow (S')$ **or** $(\perp)$:

Each $A_j \in \omega$ publicly broadcasts x_j. Using these shares they compute $X' =$

$G'(0)$. If $X' = 0$ they halt the protocol and output $\perp$. Otherwise each A_j privately computes a value $\beta_j = R \cdot y_j$. A_j then broadcasts the value:

$$\upsilon_j = \beta_j - R\left((X')^{W+2} + \sum_{l=1}^{W} S_{l_j} \cdot (X')^l\right)$$

Collectively ω then reconstructs the value υ from the shares υ_j. If $\upsilon \neq 0$ the protocol halts and the output is $\perp$, otherwise the servers reconstruct and broadcast the values $S'_1, \cdots, S'_W$ from their shares $S_{1_j}, \cdots, S_{W_j}$.

Theorem 6. *The proposed construction is a secure* (t, n, k, ϵ) *OSSCD scheme with error probability* $\epsilon = \frac{W+1}{q-1}$.

Proof. The proof of this is analogous to the proof of theorem 5 i.e., the coalitions of k servers and t participants cannot compute any information relating to S. The error probability is given as $\epsilon = \frac{W+1}{q-1}$. This is based on the error probability of the AMD code: $\epsilon = \frac{W+1}{q}$ coupled with the fact that, $X \neq 0$ (giving a $1/(q-1)$ probability of guessing X). □

3.6 Conclusion

In this Chapter we demonstrated the new concept of OSSCD which allows participants to offload the verification phase of an SSCD protocol. In a typical SSCD scheme, either the participants reconstruct the secret among themselves, or they assign the reconstruction to a single party who publicly carries out the necessary computations. In both cases, even if cheating is detected and the reconstructed secret is found to be faulty, the cheating participants are able to work backwards and obtain the original, uncorrupted secret. Our new protocol corrects this flaw by outsourcing verification of the secret to a set of servers. If verification fails then neither the servers nor the participants achieve the secret.

Unlike the solutions used for such purposes in multi-party computation, our protocol does not require private and authenticated channels between participants,

reducing both the computational and communication complexity required by the participants. This is ideally suited for participants with low computational power, as the bulk of the complexity can be abstracted away by the servers.

Chapter 4

Distributed Oblivious Polynomial Evaluation

4.1 Introduction

In this Chapter we investigate a method of achieving unconditionally secure distributed OPE (DOPE) in which the function of the sender is distributed amongst a set of n servers. Specifically, we introduce a model for DOPE based on the model for distributed oblivious transfer (DOT) described by Blundo et al. in 2002. We then describe a protocol that achieves the security defined by our model. Our DOPE protocol is efficient and achieves a high level of security. Furthermore, our proposed protocol can also be used as a DOT protocol with little to no modification.

To refresh the brief explanation of OPE given in Chapter 1, OPE was first introduced by Naor and Pinkas in 1999 [81]. An OPE protocol involves two parties, a receiver, R who holds a private value, α and a sender, S who holds a private polynomial, $f(x)$. Informally, an OPE protocol allows R to learn the evaluation of S's polynomial at his private value i.e., $f(\alpha)$, whilst keeping S from learning α and R from learning any more information about $f(x)$. A more formal definition, adapted from [22] is given below:

Definition 5. *[22] An OPE protocol is composed of two parties, S who has a polynomial $f(x)$ over a finite field $\mathbb{F}$ and R who has an input value $\alpha \in \mathbb{F}$. Correctness is achieved if, at the end of the protocol, R learns $f(\alpha)$. Security is guaranteed if the following two conditions are met after the protocol has been executed:*

1. *S cannot reduce his uncertainty of α, i.e., the probability of S computing α is*

$1/|\mathbb{F}|$.

2. *R does not learn any information relating to $f(x)$, other than $f(\alpha)$.*

OPE has been found to have a myriad of applications in such things as secure computation [53], oblivious neural learning [22], secure set intersection [63] and privacy preserving data mining [78]. As a result of this, an extensive amount of research has been conducted on this topic [22, 59, 62, 63, 75, 80, 100, 102].

To the best of the author's knowledge there exists only three unconditionally secure OPE protocols in the literature. The first unconditionally secure OPE was given by Chang and Lu [22]. To achieve information theoretic security they use a third party who takes an active role in the protocol execution. The second information theoretic secure OPE protocol was given by Hanaoka et al. in [62] (and was later expanded on in [100]). Their protocol also requires the use of a third party although, in their protocol the third party acts as an initialiser, in that they merely distribute some (unrelated, effectively random) information at the start of the protocol and then take no further part in the protocol execution. The third OPE protocol that achieves information theoretic security was given by Li et al. [75]. Their protocol takes a different approach and instead utilises a set of servers to collectively implement the function of the sender. We denote such a scheme as a distributed oblivious polynomial evaluation (DOPE) protocol, in order to differentiate this type of scheme from the other three-party protocols.

In the DOPE protocol of Li et al. [75] the sender initialises the protocol by distributing some information amongst a set of $n \geq 2$ servers. Following this, S takes no further part in the protocol. To compute his evaluation, R communicates with a subset composed of t amount of these servers where $t \leq n$ is known as the threshold. The sender's security is guaranteed against a coalition composed of $l - 1$ servers and R; whilst the receiver's privacy is guaranteed against a subset

of $b-1$ servers, where $b+1 < t \leq n$. Li et al. also show how to improve this scheme allowing for the greater threshold of $b = l = t$ by introducing some publicly known information. However, we note that this increase in security comes at a cost. Namely, it dramatically increases the overall complexity of their protocol and it also allows both R and the servers to gain some extra information about $f(x)$. The authors of [75] state that in their protocol the servers receive a set of linear equations relating to the coefficients of $f(x)$ such that (R) obtains more information simply from contacting more servers during the protocol. Specifically, like many DOT protocols, they require that there is a mechanism that limits the amount of servers (R) is allowed to contact, as contacting over this specified amount will result in (R) gaining (S)'s polynomial. The protocol introduced in this chapter does not suffer from such a flaw and allows (R) to contact as many servers as possible.

4.1.1 Our Contribution

In this Chapter we develop such a protocol by first describing a model of DOPE and then introducing an efficient DOPE protocol that achieves the security defined in our model. Specifically, our proposed protocol allows R to compute his evaluation by simply broadcasting some information and then receiving contact from $t+1$ or more servers. The protocol achieves security for R against a coalition of t servers and security for S against a coalition composed of l servers and R and does not leak any information relating to either $f(x)$ or α.

To develop a model of DOPE we simply apply a slightly modified version of the already established and well studied security framework developed by Blundo et al. [14, 13] for the purpose of distributed oblivious transfer (DOT) [6, 25, 35, 34, 82, 84]. We then give the construction of a DOPE protocol that is secure under this model. An interesting property of our protocol is that it can also be utilised as a DOT protocol with little to no modification.

Our protocol achieves security equivalent to what Blundo et al. describe as a strong DOT protocol [13]. That is, our DOPE protocol is secure against a coalition composed of t servers and R even after R has received $f(\alpha)$.

4.2 Model

Similar to a DOT protocol a DOPE protocol consists of a sender, S, the receiver, R and n servers, $s_1, \cdots, s_n$. As per Definition 5 the sender has a polynomial, $f(x)$ of degree $k \geq 1$ over $\mathbb{F}$, whilst the receiver has a point $\alpha \in \mathbb{F}$, such that $|\mathbb{F}| = q$ where q is a prime number and $q > \max(k, n)$. We assume a standard model of communication present in many multi-party protocols [7] i.e., a synchronous broadcast connection exists between the servers and R, such that R can privately and simultaneously send the same message to all of the servers. Additionally, we assume each server has a secure channel that allows them to send private messages to R. DOPE consists of two phases:

1. **Initialisation:** S privately distributes some information relating to $f(x)$ to each of the n servers. Following this S takes no further part in the protocol.

2. **Evaluation:** R broadcasts some information to all of the servers. A set of $t + 1$ or more servers send a response to R who then uses this information to compute $f(\alpha)$.

In order to achieve both correctness and security a DOPE protocol must satisfy the following security conditions, originally given by Blundo et al. [13] and informally stated by Corniaux and Ghodosi [34] for the purpose of DOT:

1. ***Correctness:*** R is able to compute the requested evaluation after receiving information from $t + 1$ servers.

2. ***Receiver's Privacy:*** A coalition of t servers cannot compute any information relating to α.

3. ***Sender's Privacy:*** After the initialisation phase (but before the evaluation phase) a coalition composed of t servers and R cannot compute any information relating to $f(x)$.

4. ***Sender's Privacy After Protocol Execution:*** After communication between R and the servers has occurred and R has computed $f(\alpha)$, a coalition composed of t servers and R cannot compute any information relating to $f(x)$; other than what the evaluation of R's chosen value (i.e., $f(\alpha)$) has already revealed.

In our model we assume that all participants follow the protocol exactly, i.e., they are semi-honest. A benefit of our model is that the degree of $f(x)$ (given as k) is not related to the threshold parameter, t. This allows for a flexible and easily changeable level of security. For instance, even if the degree of k is small S can ensure security against a large number of servers by assigning a high value to t.

In regards to the security conditions given by Blundo et al. it was shown that a DOT protocol that achieves all four security conditions could only be achieved in two rounds of communication between the servers and R, or by allowing S to contact R during the initialisation phase. This also proves true for our DOPE protocol which is given in the next section. We note that, similar to Blundo's "Strong DOT Protocol" [13] our protocol assumes that S correctly distributes the information to the servers and does not try to initiate any further contact with R or the servers after the initialisation phase. The rationale behind this assumption (which has long been held in the field of DOT) is that the sender exists much like the dealer in a secret sharing scheme i.e., they are a third party that exists solely to distribute their respective information then take no further part in the protocol.

Our model of DOPE is very similar to the model given by Li et al. [75], we have merely extended their somewhat brief explanation by utilising the tools Blundo et al. developed for DOT.

4.3 DOPE Protocol

In this section we describe our DOPE protocol and then evaluate the security of the protocol against the security conditions given in the previous section.

In our proposed protocol S utilises Shamir's secret sharing scheme [97] to securely distribute his polynomial among the n servers. This has been covered in previous Chapters and, as such, is omitted here.

4.3.1 The Proposed DOPE Protocol

The underlying idea behind our protocol is similar to the protocol given by Li et al. [75], in that we have S utilise Shamir's secret sharing scheme to distribute shares of the coefficients of $f(x)$ to each server.

To achieve privacy for R we have S distribute some semi-random information along with the shares of the coefficients. Each server receives shares of this information whilst R receives the information in its entirety. Using the distributed information R can then easily distribute his value α among the servers, who then perform a computation and send the output back to R. Following this, R computes a polynomial of which the free term is his desired evaluation.

The actual method utilised to distribute shares of α was originally given in [48] as a means to securely introduce input values under a shared MAC key in multi-party computation. We specifically use it to allow the contacted servers to efficiently compute a share of α multiplied by a given coefficient of $f(x)$. The full DOPE protocol is given in figure 4.1.

Input: S has the polynomial $f(x) = a_0 + a_1 x + \cdots + a_k x^k$ and R the value α.

Output: R receives $f(\alpha)$ and S gets nothing.

Initialisation

1. S creates a set of random values $r_1, \cdots, r_k$ and computes k values of the form $\gamma_i = r_i \cdot a_i$ for $i = 1, \cdots, k$.

2. For each coefficient, a_h $(h = 0, \cdots, k)$, S computes a random polynomial, $A_h(x)$ of degree at most t such that $A_h(0) = a_h$. He does the same for each γ_i value, computing k polynomials of the form $\Gamma_i(x)$ with free term $\Gamma_i(0) = \gamma_i$.

3. Using Shamir's secret sharing scheme S distributes these values among the servers, giving server s_j $(j = 1, \cdots, n)$ the following information:

   - k shares of the form $\gamma_{i_j} = \Gamma_i(j)$
   - $k + 1$ shares of the form $a_{h_j} = A_h(j)$

4. S privately sends to R the values $r_1, \cdots, r_k$ and then takes no further part in the protocol.

Evaluation

1. R broadcasts to all servers a set of k values of the form $\epsilon_i = \alpha^i - r_i$.

2. A set of $t + 1$ or more servers, denoted as $\mathcal{W}$ respond to R's broadcast values. Each server, $s_j \in \mathcal{W}$, computes and sends to R the share:

$$z_j = a_{0_j} + \sum_{i=1}^{k} (a_{i_j} \cdot \epsilon_i + \gamma_{i_j})$$

3. R performs Lagrange interpolation across each z_j value to compute the polynomial $Z(x)$ with free term $Z(0) = f(\alpha)$.

Figure 4.1 : The Proposed DOPE Protocol

In section 4.2 we stated the result of Blundo et al. [13] which proved that a strong DOT protocol can only be achieved in two rounds. The same is true for our DOPE protocol, we merely circumvented this limitation by allowing S to contact R in the initialisation phase. Specifically, in our protocol we have S directly send the values $r_1, \cdots, r_k$ to R in the initialisation phase. This is actually not strictly necessary, and to limit direct contact between S and R we could instead have S distribute shares of $r_1, \cdots, r_k$ to each server. At the start of the evaluation phase a set of $t+1$ or more servers would then send R their shares of these values. This results in a two round protocol in which R only has to be present during the evaluation phase. This is, of course, the exact same approach taken by Blundo et al. [13] for their strong DOT protocol.

In fact, due to the similarity of the models our DOPE protocol can easily be converted to a strong $\binom{1}{m}$ DOT protocol. In a $\binom{1}{m}$ DOT protocol the receiver wishes to learn 1 of m secrets held by S. If we define S's secrets as $\omega_1, \cdots, \omega_m$ then we can achieve DOT by having S compute $f(x)$ so that $f(i) = \omega_i$ for $i = 1, \cdots, m$. To learn the i^{th} secret R sets $\alpha = i$ and then executes the rest of the protocol as before.

4.3.2 Evaluation

In this section we evaluate the security of the proposed DOPE protocol by proving that it meets the conditions given in section 4.2.

Correctness

Theorem 7. *If all participants follow the protocol correctly the receiver obtains $f(\alpha)$ by contacting $t+1$ servers.*

Proof. At the end of the evaluation phase R will have received $t+1$ shares of the

form:

$$z_j = a_{0_j} + \sum_{i=1}^{k} (a_{i_j} \cdot \epsilon_i + \gamma_{i_j})$$

Where the share z_j is from server s_j. Due to the homomorphic nature of Shamir's secret sharing scheme, linear operations performed on shares also correspond to the secrets and polynomials these shares are computed from [8]. In other words the shares correspond to the polynomial:

$$Z(x) = A_0(x) + \sum_{i=1}^{k} (A_i(x) \cdot \epsilon_i + \Gamma_i(x))$$

The free term of each $A_i(x)$ is $A_i(0) = a_i$, similarly $\Gamma_i(0) = r_i \cdot a_i$, therefore:

$$Z(0) = a_0 + \sum_{i=1}^{k} (a_i \cdot \epsilon_i + r_i \cdot a_i)$$

Since $\epsilon_i = \alpha^i - r_i$ this becomes:

$$\begin{aligned} Z(0) &= a_0 + \sum_{i=1}^{k} (a_i \cdot \alpha^i - a_i \cdot r_i + r_i \cdot a_i) \\ &= a_0 + \sum_{i=1}^{k} a_i \cdot \alpha^i \\ &= a_0 + a_1 \cdot \alpha + a_2 \cdot \alpha^2 + \cdots + a_k \cdot \alpha^k \\ &= f(\alpha) \end{aligned}$$

□

Receiver's Privacy

Theorem 8. *A coalition of t servers cannot compute any information relating to α.*

Proof. Suppose that a set of t servers, who were all contacted by R, form a coalition. The goal of this coalition is to compute some information relating to α. Collectively the servers have a set of t shares relating to each coefficient of $f(x)$, (i.e. $a_0, \cdots, a_k$) as well as t shares relating to the product of each random value and a coefficient,

i.e. $\gamma_i = a_i \cdot r_i$ for $i = 1, \cdots, k$. Additionally, the servers also have k values of the form $\epsilon_i = \alpha^i - r_i$ which gives the following system of equations:

$$\epsilon_1 = \alpha - r_1$$

$$\epsilon_2 = \alpha^2 - r_2$$

$$\vdots$$

$$\epsilon_k = \alpha^k - r_k$$

From the above system, we can see that to compute α the coalition would first need to compute a given r_i value. However, due to the perfectly secure nature of Shamir's secret sharing scheme [36, 97], t shares does not reveal any information relating to a given secret. As a result of this, the coalition of servers cannot compute any information relating to any of the coefficients of $f(x)$, the γ_i or the r_i values. Since each r_i value is chosen at random, and they cannot compute any information relating to these values the above system is composed of k independent equations and $k+1$ unknowns (each r_i value in addition to α) which results in every possible value of α being equally likely. □

Sender's Privacy

Theorem 9. *A coalition composed of t servers and R cannot compute any information relating to $f(x)$ during initialisation.*

Proof. At the end of the initialisation phase a coalition of t servers and R will have the following information:

1. The values $r_1, \cdots, r_k$
2. t shares corresponding to each coefficient polynomial $(A_0(x), \cdots, A_k(x))$, which gives $t(k+1)$ shares.

3. t shares relating to the each of other set of polynomials $(\Gamma_1(x), \cdots, \Gamma_k(x))$, giving kt collective shares.

As per the proof of Theorem 7 it is impossible to compute any information about a given polynomial, of degree t, with only t shares. However, the free term of each polynomial of the form $\Gamma_i(x)$ for $1 = 1 \cdots k$ is $\Gamma_i(0) = r_i a_i$ where r_i is a known quantity. The coalition can use this knowledge to compute a polynomial with free term a_i. This allows them to hold two polynomials with the free term a_i.

We note that even with this extra knowledge they cannot achieve anything as a_i is unknown to them and furthermore, holding two sets of t shares relating to two different polynomials with the same free term does not actually reveal any information [70, 97]. □

Sender's Privacy After Protocol Execution

Theorem 10. *A coalition composed of t servers and R cannot compute any information relating to $f(x)$ after the execution of the protocol, other than what the evaluation of R's chosen value, $f(\alpha)$, gives them.*

Proof. The proof of this is analogous to the previous proof with the addition of some extra information, namely the information given to R by the other servers who contacted him. For the sake of the proof we will assume that the first $t + 1$ servers contact R. Without loss of generality and for the sake of convenience, assume that the coalition is composed of R and the first t servers, $s_1, \cdots, s_t$. This coalition has the exact same information as before, this time however, they also have the added knowledge of the other (honest) server's responses to R. That is:

$$z_{t+1} = a_{0_{t+1}} + \sum_{i=1}^{k} (a_{i_{t+1}} \cdot \epsilon_i + \gamma_{i_{t+1}})$$

If the coalition are able to compute any of the polynomials used to distribute the coefficients of the senders polynomial, $A_0(x), \cdots, A_k(x)$, or even the polynomials used to distribute the product of the random values and the coefficients, $\Gamma_1(x), \cdots, \Gamma_k(x)$, then they can easily compute the value of a given coefficient of $f(x)$. We must therefore prove that this is not possible.

First, let $h = 0, \cdots, k$ and let $i = 1, \cdots, k$ then any given server, s_j, contacted by R has $k+1$ shares of the form a_{h_j} corresponding to $A_h(x)$ and k shares of the form γ_{i_j} corresponding to $\Gamma_h(x)$. We can write these polynomials as:

$$A_h(x) = a_h + A_{h_1}x + A_{h_2}2x^2 + \cdots + A_{h_t}x^t$$

$$\Gamma_i(x) = r_i a_i + G_{i_1}x + G_{i_2}x^2 + \cdots + G_{i_t}x^t$$

Using this notation the response of each server, z_j for $j = 1, \cdots t+1$, can be written as:

$$z_j = \sum_{y=1}^{k} a_y \alpha^y + \sum_{h=0}^{k}\left(\epsilon_h\left(\sum_{v=1}^{t} A_{h_v} j^v\right)\right) + \sum_{i=1}^{k}\left(\sum_{v=1}^{t} G_{i_v} j^v\right)$$

Therefore, from $t+1$ responses R obtains a system composed of $t+1$ equations and $(t+1)(k+1) + kt$ unknowns, specifically, $t+1$ unknowns from each of the $k+1$ polynomials of the form $A_h(x)$ and t unknowns from each of the k amount of polynomials of the form $\Gamma_i(x)$.

Each z_j is composed of a linear combination of polynomials of degree t. Therefore, the system that R constructs is composed of $t+1$ independent equations. However, we note that $t \geq 1$ and $k \geq 1$, meaning that the amount of unknowns will always be greater than the amount of independent equations. As a result of this, R and the coalition of t servers cannot compute anything from just the responses.

In fact, even with the direct shares of each of the t servers in the coalition they still cannot compute any information. This is because the equation used to describe a given share is not linearly independent to the equation used for a given z_j i.e. each z_j is simply a linear combination of a given participant's share and thus, is not a

separate (independent) equation.

The net result for the coalition is a system composed of only $t + 1$ independent equations and $(t + 1)(k + 1) + kt$ unknowns, resulting in each value of a given coefficient of $f(x)$ being equally likely. □

Chapter 5

Unconditionally Secure Oblivious Polynomial Evaluation: A Survey and New Results

5.1 Introduction

In this Chapter we review some of the myriad of applications OPE has been utilised in and, additionally, take an in depth look at the special case of information theoretic OPE.

Specifically, we provide a current and critical review of the existing information theoretic OPE protocols in the literature. We divide these protocols into two distinct cases (three party and distributed OPE) allowing for the easy distinction and classification of future information theoretic OPE protocols. In addition to this work we also develop several modifications and extensions to existing schemes, resulting in increased security, flexibility and efficiency. Lastly, we demonstrate the flaws in a previously published OPE scheme .

An extensive amount of research has been conducted on OPE since its introduction [30, 59, 65, 102, 75, 83, 100]. Of particular interest are the many applications and uses which researchers have found for OPE.

This valuable tool has been used in protocols such as multi-party computation (MPC) [31, 53], secure mean computation [91], oblivious neural learning [22, 23], oblivious keyword search [88], and privacy preserving data mining [76], to name a few. In fact, OPE is an integral part of many protocols utilised in modern cryptography. In general we can divide these protocols into the two categories, previously discussed in Chapter 1:

1. Multi-Party Computation (MPC) Protocols : MPC allows a set of n participants to securely compute any given function over their privately held information. More formally, a set of n participants, $P_1, \cdots, P_n$ with respective private inputs, $x_1, \cdots, x_n$ can compute a given function $f(x_1, \cdots, x_n)$ without revealing any information relating to their inputs.

2. Privacy Preserving Protocols: We choose to use this term to refer to protocols that solve a specific function or problem, with the same level of privacy utilised in MPC. We can actually view these sorts of protocols, and OPE itself, as a subset of MPC protocols.

5.1.1 OPE and Multi-Party Computation

Many of the recently proposed MPC protocols within the literature [45, 48, 53, 72] utilise OPE as an offline protocol for the generation of correlated random data. To clarify this statement, in such protocols an MPC is split into two phases:

1. **Offline Phase:** In this phase participants compute some effectively random and shared data.

2. **Online Phase:** This phase is where participants are able to compute a function across their private data. The MPC protocol can be computed efficiently using the correlated random data computed in the offline phase.

Using OPE in such a fashion allows for a fast online phase, resulting in an efficient MPC protocol.

The other use that OPE has seen in MPC is as a multiplication protocol in the online phase. We have shown [31] (Chapter 6) that a certain type of multiplication can be computed efficiently and securely by utilising a modified OPE protocol based on the work of Tonicelli et al. [100].

5.1.2 OPE and Privacy Preserving Protocols

As stated previously, a privacy preserving protocol can be seen as a subset of MPC. These types of protocols compute a specific function with great efficiency, many such privacy preserving protocols utilise OPE as an essential building block.

Lindell et al. [79] utilised OPE as an integral part of their secure data mining protocol, which allows participants to securely run standard data mining algorithms across their privately held information. Similar to this, Chang et al. [22, 23] utilised OPE for the purpose of oblivious neural learning i.e., training a neural network across private data. In [64] Hazay showed how a set of participants could securely compute the intersection of their privately held sets. Ogata and Kurosawa [88] utilised OPE to develop an oblivious keyword protocol, wherein a participant can search among a secure database whilst keeping the information s/he was searching for private. Lastly, a secure voting scheme was developed from OPE in [90] by Otsuka and Imai.

5.1.3 Outline and Contribution

It is evident that OPE is a valuable protocol that has many applications and uses. However, to the best of the author's knowledge there has not yet been any surveys or reviews published on this deeply interesting topic. We seek to rectify this by presenting a thorough review of a unique type of OPE protocol. Namely, as with all of the protocols in this thesis, we focus on the specific case of information theoretic (or unconditionally secure) OPE, wherein it is assumed each of the participants (and any given adversary) has unlimited computational resources (see section 5.2 for more information).

In particular, we review the current results present within the literature and then modify some of these protocols to gain improvements in efficiency, flexibility and security. To summarise, our results are threefold:

1. We provide a thorough description and critical review of currently known information theoretic OPE protocols. Furthermore, we show that each of the information theoretic OPE protocols within the literature can be classified under two sub-fields, three party OPE and distributed OPE.

   This result, or classification, directly corresponds to the already well known and researched area of information theoretic oblivious transfer [35, 43, 95].

2. We do not merely describe and review each protocol. We also develop modified versions of specific protocols, extending their capabilities, efficiency and security.

   Additionally, we further prove the link between OT and OPE by demonstrating that a previously published distributed oblivious transfer (DOT) protocol can easily be adapted to a distributed OPE protocol.

3. Lastly, we show that a previously published 'unconditionally secure OPE' scheme does not, in fact, achieve unconditional security.

The rest of this Chapter is organised as follows. Section 5.2 provides some background on information theoretic OPE. Sections 5.3 and 5.4 investigate the two distinct categories of information theoretic OPE, reviewing current results and also describing our own research in this field. Section 5.5 examines the OPE scheme shown in [15] and demonstrates that it is not secure. Finally, section 5.6 concludes the Chapter.

5.2 Background

An information theoretic OPE is a two party protocol that is secure against an adversary, or participants, who have unlimited computational power and resources. This is more formally defined below.

Definition 6. *Given definition 5 (Chapter 4), we define the output of an OPE protocol (i.e., the value the receiver, $\mathcal{R}$ computes as his/her desired evaluation) as f_α. Let A define the set of all possible evaluation points such that $\alpha \in A$ and let E be the set of all possible evaluations such that $f_\alpha \in E$. Lastly, let $V_\mathcal{R}$ denote $\mathcal{R}$'s view (i.e., all information known and held by $\mathcal{R}$) upon completion of the protocol, and let $V_\mathcal{S}$ denote $\mathcal{S}$'s view. Assuming that all participants are honest, an OPE protocol obtains information theoretic security if the following conditions hold.*

- *Correctness.*

$$Pr[f_\alpha = f(\alpha)] = 1$$

- *Security for $\mathcal{R}$. Given $V_\mathcal{S}$ defines $\mathcal{S}$'s view of the distribution of any given possible value of α and define $\alpha' \in A$ as another possible value for α chosen by $\mathcal{R}$. We say that security is maintained for $\mathcal{R}$ if (from the view of $V_\mathcal{S}$) it is indistinguishable when $\mathcal{R}$ chooses α or $\mathcal{R}$ chooses α'.*

- *Security for $\mathcal{S}$. Similarly, given that $\mathcal{R}$ has obtained the evaluation point $f_\alpha = f(\alpha)$, define $g_\alpha = g(\alpha)$, where $g(x)$ is another possible polynomial chosen by $\mathcal{S}$. Security is maintained for $\mathcal{S}$ if, from the point of view of $V_\mathcal{R}$, f_α and g_α are statistically indistinguishable from each other.*

The above definition is a more expanded version of Definition 5 that formalises the security requirements for OPE. Put simply it states that, if all participants are honest then the value computed by $\mathcal{R}$ at the end of the protocol will be equal to $f(\alpha)$. In terms of security, Definition 6 states that upon completion of the OPE, $\mathcal{S}$ cannot reduce their uncertainty of $\mathcal{R}$'s evaluation point (α) and $\mathcal{R}$ cannot reduce their uncertainty of $\mathcal{S}$'s polynomial, i.e., for any given input all possible outputs are equally likely.

Computationally secure protocols, need not rely on such stringent measures of

security. Instead, in a computationally secure protocol, security is assured if $\mathcal{S}$ or $\mathcal{R}$ can reduce their uncertainty of (respectively) α and $f(x)$ only by expending some (defined) limit of computational power and/or time. For example, an OPE scheme could be considered computationally secure if $\mathcal{R}$ could reduce his/her uncertainty of $f(x)$ in exponential time only.

We note that, although information theoretic protocols have a far higher level of security, this comes with a trade off in communication complexity and the number of participants. Specifically, most purely information theoretic OPE protocols within the literature tend to have a high communication complexity. The upside to this, however is that information theoretic protocols are often computationally efficient. Additionally, it has long been understood that it is not possible to have information theoretic security with only two participants [28, 38].

This very statement seems to preclude any possibility of an information theoretic OPE protocol. However, numerous researchers have cleverly avoided this conundrum by introducing a third party (or a set of third parties) who takes part in, but gains no information from the OPE protocol. We specifically refer to what we dub as both three-party OPE and distributed OPE which are (informally) described below.

1. **Distributed OPE (DOPE)**. A DOPE protocol consists of $n+2$ participants, the sender and receiver, along with n servers. In this type of protocol $\mathcal{R}$ and $\mathcal{S}$ compute the OPE by communicating with n servers. At the start of the protocol $\mathcal{S}$ distributes some information among the servers. Later, $\mathcal{R}$ contacts a subset of these servers, who provide him/her with enough information to compute his/her evaluation.

   As before, both the evaluation point and the polynomial should remain private, and furthermore a coalition of servers should not be able to compute anything related to either $\mathcal{R}$ or $\mathcal{S}$'s private information. An additional requirement

is that a coalition composed of a subset of servers and $\mathcal{R}$ cannot compute anything related to $\mathcal{S}$'s polynomial.

The benefit of this type of protocol is that after $\mathcal{S}$ distributes their data, they need not (and is not expected to) take any further part in the protocol. This provides a high level of availability for $\mathcal{R}$ in that he/she is able to compute his/her OPE at any given time, without waiting on $\mathcal{S}$. This is further improved when we consider that $\mathcal{R}$ need only contact a subset of servers; thus if some servers are not available $\mathcal{R}$ may still compute their evaluation.

DOPE also has a requirement on the security of the sender and receiver in that maximum security can only be achieved for one of these individuals. As evidenced in [75] we require that $\gamma_1 + \gamma_2 \leq t$ where γ_1 and γ_2 are potential server collusion sizes for both $\mathcal{R}$ and $\mathcal{S}$ respectively.

2. **Three-Party OPE (TOPE)** A TOPE protocol involves a single third party who takes part in the protocol alongside $\mathcal{S}$ and $\mathcal{R}$. This third party is mutually trusted and provides information to both participants which allow them to efficiently and securely compute an OPE.

   As with the servers in DOPE, the third party should not be able to compute any information related to $\mathcal{S}$'s polynomial or $\mathcal{R}$'s evaluation point. However, unlike DOPE, it is expected that the third party will not actively try to cheat by sharing (private) information with either $\mathcal{S}$ or $\mathcal{R}$. We note that TOPE protocols are far more efficient than DOPE protocols and often a lot simpler and easier to understand.

In this work, we examine information theoretic OPE schemes that are secure in the presence of a semi-honest (AKA honest but curious) adversary. This assumes that all of the participants will follow the protocol exactly, but will try to learn as much extra information as possible. In the case of DOPE this means that coalitions

of servers will attempt to compute some information, whilst in TOPE we assume that no coalitions are formed, rather both $\mathcal{R}$ and $\mathcal{S}$ will individually attempt to compute information related to both α and $f(x)$ (whichever they themselves do not directly know).

Within the literature, there currently exist very few information theoretic OPE protocols. However, the usefulness of these protocols (as seen in section 5.1) is not in doubt. TOPE has been used in both MPC and privacy preserving protocols, whilst DOPE draws strong parallels to distributed oblivious transfer (DOT) which is a well established and thoroughly researched field with many rich applications and protocols.

Thus, it is our hope to further illuminate information theoretic OPE by reviewing the current protocols and providing our own research in this field. In the following sections we formally define DOPE and TOPE and investigate the protocols present within the literature.

5.3 Distributed OPE

To the best of the author's knowledge there exists only two DOPE protocols within the literature, the 'distributed oblivious function evaluation' (DOFE) of Li et al. [75] and the DOPE protocol reviewed in the previous Chapter [30]. Although both DOPE protocols in the literature differ in their security requirements and definitions, we can provide a blanket model that suffices to broadly explain the requirements of both DOPE protocols. We will, of course, specify the specific requirements of each DOPE protocol before examining them. Our broad model and security requirements of a DOPE protocol are given in the preceding Chapter.

5.3.1 The DOFE Protocol

Li et al. [75] describe a set of three DOPE protocols, each with varying levels of security and flexibility. For our purposes, we examine their second scheme, however in section 5.3.3 we show that a sub-protocol for a DOT scheme shown in [25] can be slightly altered to produce the first DOPE scheme described by Li et al. In the DOPE protocol of Li et al. [75] the sender's security is guaranteed against a coalition composed of $l-1$ servers and $\mathcal{R}$. Whilst the receiver's privacy is guaranteed against a subset of $b-1$ servers and $\mathcal{S}$, such that $b+l < t \leq n$ where the security of both $\mathcal{R}$ and $\mathcal{S}$ is guaranteed against a coalition composed of $t-1$ or less servers. The full protocol is given in Figure 5.1. This is a flexible scheme that can be easily altered to suit a given environment. However, it is evident that increasing the security or privacy threshold for $\mathcal{S}$ would result in a decrease of security for $\mathcal{R}$ and vice versa. Li et al. show how to avoid this, achieving a threshold of $b = l = t$, unfortunately though this increase in security comes with a cost complexity (as discussed in Chapter 4).

Besides the overall complexity of the scheme increasing, the security modifications also allow $\mathcal{R}$ to learn extra information about $f(x)$ (see Chapter 4). Although this information is not enough for $\mathcal{R}$ to compute anything in an isolated setting, it does mean that this scheme may not be suitable for implementation as part of a larger protocol.

The only other DOPE protocol present within the literature is the scheme devised in Chapter 4. We note that this is a robust scheme in which $\mathcal{R}$ simply broadcasts (publicly) some information to which a set of servers then respond. This means that $\mathcal{R}$ does not necessarily need to pick the specific servers he communicates with, rather the servers can either all respond or just have a minimum subset of required servers respond.

Input: $\mathcal{S}$ has the polynomial $f(x) = a_0 + a_1x + \cdots + a_kx^k$ and $\mathcal{R}$ the value α.

Output: $\mathcal{R}$ receives $f(\alpha)$ and $\mathcal{S}$ gets nothing.

Initialisation

1. $\mathcal{S}$ selects and broadcasts $k+1$ random, distinct values: $x_0, x_1, \cdots, x_k$.
2. Using these values $\mathcal{S}$ privately computes $y_0, y_1, \cdots, y_k$ such that $y_i = f(x_i)$ where $i = 0, \cdots k$.
3. Next, $\mathcal{S}$ computes $k+1$ random polynomials, $f_0(x), f_1(x), \cdots, f_k(x)$ where $f_0(0) = y_0, f_1(0) = y_1 - y_0, \cdots, f_k(0) = y_k - y_0$ such that the degree of $f_0(x)$ is at most $t-1$ and the other k polynomials have degree at most $l-1$.
4. $\mathcal{S}$ send to each server, s_j the share $A_j = (f_0(j), \cdots, f_k(j))$.

Evaluation

1. $\mathcal{R}$ computes the random values $d_1, \cdots, d_k$ such that they satisfy $\alpha = d_1\mathbf{x_1} + \cdots + d_k\mathbf{x_k}$. Where, for any value x, the value $\mathbf{x}$ denotes $(1, x, x^2, \cdots, x^k)$.
2. $\mathcal{R}$ then uses these values to compute a set of random polynomials, $Q_1(x), \cdots, Q_k(x)$ of degree at most $b-1$, where $Q_i(0) = d_i$ for $i = 1, \cdots, k$.
3. $\mathcal{R}$ selects a subset of t servers denoted as ω. He then sends to each $s_j \in \omega$ the values $B_j = (1, Q_1(j), \cdots Q_k(j))$.
4. Each $s_j \in \omega$ computes and sends to $\mathcal{R}$ the value $R(j) = \langle A_j, B_j \rangle$ i.e., the inner (dot) product of the two vectors.
5. $\mathcal{R}$ computes $f(\alpha)$ by interpolating over the set of $R(j)$ values he received to compute the polynomial $R(x)$. He takes $R(0)$ as his evaluation.

Figure 5.1 : DOFE Protocol [75]

5.3.2 Our Proposed DOPE protocol

Whilst the security benefits of our previously described DOPE scheme are obvious, it is not without flaws. For instance, this protocol requires $\mathcal{S}$ to communicate directly with $\mathcal{R}$ during the initialisation stage. Although we show how this case can be fixed (by simply assigning shares of the required values to the servers, who can then share it with $\mathcal{R}$), the result is an increase in communication complexity.

Furthermore, and perhaps more alarmingly, the protocol requires (like many previously published DOT protocols) that $\mathcal{S}$ not communicate with any of the servers after the initialisation phase. It is easy to see that if this were to occur it would be a trivial matter for $\mathcal{S}$ to compute the exact value of α.

Specifically, all it would take is one server revealing to $\mathcal{S}$ the value of ϵ_i. This would allow $\mathcal{S}$ to compute $\alpha = \sqrt[i]{\epsilon_i + r_i}$. Resulting in a complete loss of privacy for $\mathcal{R}$.

Unfortunately this predicament is an inherent problem present in all such schemes that achieve the maximum level of security. As noted by Cheong et al. [25], achieving this level of security in a DOT protocol also results in the same issue. To overcome this problem within the field of DOT, the author's of [25], developed a robust and flexible DOT scheme that achieved what they described as the maximum security a DOT scheme could possibly achieved. One in which the security issue highlighted above does not exist.

In the next section we show that an interesting sub protocol used in their DOT scheme can be modified to provide a secure DOPE protocol, with the flexible security thresholds of the DOFE protocol. In fact, we show that this sub-protocol can be adapted into the first DOPE protocol described in [75] by Li et al. with minimal effort.

Input: $\mathcal{S}$ has the polynomial $f(x) = a_0 + a_1 x$ and $\mathcal{R}$ the value α.

Output: $\mathcal{R}$ receives $f(\alpha)$ and $\mathcal{S}$ gets nothing.

Initialisation

1. $\mathcal{S}$ computes two random polynomials, $B_0(x)$ and $B_1(x)$, such that $B_0(x)$ is of degree at most t with $B_0(0) = a_0$ and $B_1(x)$ is of degree at most γ_2 with $B_1(0) = a_1$. He combines these two polynomials to compute the bivariate polynomial $Q(x, y) = B_0(x) + B_1(x)y$.
2. Each server, s_j (for $j = 1, \cdots, n$) receives from $\mathcal{S}$ the values $(B_0(j), B_1(j))$.

Evaluation

1. $\mathcal{R}$ computes the random polynomial $S(x)$, of degree at most γ_1, where $S(0) = \alpha$.
2. Each server, s_j receives from $\mathcal{R}$ the value $S(j)$.
3. Let $Q(x, S(x)) = R(x)$, then a set of $t + 1$ or more servers (denoted by s_j) computes and sends to $\mathcal{R}$ the value $R(j) = B_0(j) + B_1(j)S(j)$.
4. $\mathcal{R}$ interpolates over these values to compute $R(x)$. Taking the value $R(0) = B_0(0) + B_1(0)\alpha = f(\alpha)$ as his desired evaluation.

Figure 5.2 : The DOLE Protocol given in [25]

5.3.3 Flexible DOPE from DOT

The security parameters of the DOT protocol devised by Cheong et al. [25] operate in much the same fashion as the DOFE protocol, in that security is guaranteed against two different thresholds. Put simply, $\mathcal{R}$'s privacy is guaranteed against a group consisting of γ_1 servers and $\mathcal{S}$, whilst $\mathcal{S}$'s privacy is guaranteed against a set of γ_2 servers and $\mathcal{R}$, where $\gamma_1 + \gamma_2 < t + 1 \leq n$. Furthermore, a group of t or less servers cannot compute anything relating to either $\mathcal{S}$'s or $\mathcal{R}$'s private information.

In this section we show how to adapt a sub protocol of their DOT scheme, in order to produce a DOPE protocol with exactly the same highly desirable security guarantees.

The core building block of this DOT scheme (the aforementioned sub-protocol, given in Figure 5.2) can be viewed as a special case of a DOPE protocol, in which the degree of $\mathcal{S}$'s polynomial is 1. Such a scheme is commonly called an oblivious linear evaluation (OLE), thus, in our case, the sub-protocol is essentially a distributed OLE (DOLE).

To create a DOPE from this DOLE protocol we simply extend the protocol, substituting the bivariate polynomial, $Q(x, y)$ with a multivariate polynomial $Q(x, y_1, \cdots, y_k)$. The full DOPE protocol is given in Figure 5.3.

Interestingly enough, the resulting protocol is exactly equivalent to the first of the three DOPE protocols given in [75] by Li et al. (as such, we point the reader to [75], for a security proof of this protocol). In fact, we note that both of these schemes (the DOT protocol of [25] and the DOPE protocol of [75]) actually utilise techniques given in the seminal work of Naor and Pinkas [82] which first introduced DOT. This result clearly shows the relationship between DOPE and DOT and establishes DOPE as an interesting field in its own right.

In the next section we examine the existing TOPE protocols within the literature.

Input: $\mathcal{S}$ has the polynomial $f(x) = a_0 + a_1x + \cdots + a_kx^k$ and $\mathcal{R}$ the value α.

Output: $\mathcal{R}$ receives $f(\alpha)$ and $\mathcal{S}$ gets nothing.

Initialisation

1. $\mathcal{S}$ computes $k+1$ random polynomials, $B_0(x), \cdots, B_k(x)$, such that $B_0(x)$ is of degree at most t with $B_0(0) = a_0$ and $B_1(x), \cdots, B_k(x)$ are of degree at most γ_2 with $B_i(0) = a_i$, for $i = 1, \cdots, k$. As before, he combines these polynomials to compute the Multivariate polynomial $Q(x, y_1, \cdots, y_k) = B_0(x) + B_1(x)y_1 + \cdots + B_k(x)y_k$.

2. Each server, s_j (for $j = 1, \cdots, n$) receives from $\mathcal{S}$ the values $(B_0(j), B_1(j), \cdots, B_n(j))$.

Evaluation

1. $\mathcal{R}$ computes the random polynomials $S_1(x), \cdots, S_k(x)$, of degree at most γ_1, where $S_j(0) = \alpha^j$.

2. Each server, s_j receives from $\mathcal{R}$ the values $(S_1(j), \cdots, S_k(j))$.

3. Let $Q(x, S_1(x), \cdots, S_k(x)) = R(x)$, then a set of $t+1$ or more servers (denoted by s_j) computes and sends to $\mathcal{R}$ the value

$$R(j) = B_0(j) + B_1(j)S_1(j) + \cdots + B_k(j)S_k(j)$$

4. $\mathcal{R}$ interpolates over these values to compute $R(x)$. Taking the value $R(0) = B_0(0) + B_1(0)\alpha = f(\alpha)$ as his desired evaluation.

Figure 5.3 : Flexible DOPE Protocol from DOT [25]. Equivalent to the first DOPE protocol given in [75].

5.4 Three-Party OPE

TOPE substitutes the n servers of DOPE for just one extra participant who takes part in the protocol. As with the servers in DOPE, this third participant

should learn nothing relating to either $f(x)$ or α, furthermore it is expected that the third party not actively collaborate with either $\mathcal{R}$ or $\mathcal{S}$.

The major benefit that a TOPE protocol has over a DOPE protocol is the need for only one extra participant. This drastically cuts down on communication complexity as there is no need to send/receive messages from a large set of servers. The downside to TOPE, is of course, that there is a central point of failure. To clarify, if the third party is compromised and/or corrupted in some way and freely shares information with either $\mathcal{S}$ or $\mathcal{R}$, then all security is lost. Another issue is that of availability, the third party must actually be present throughout the protocol, if there is no third party whom both participants are willing to trust then the OPE cannot be computed. DOPE overcomes these issues by essentially spreading out the function of the third party among the set of n servers, achieving security and availability at a far higher cost to efficiency (namely, efficiency of communication).

Of the two TOPE protocols reviewed in this section, the first uses the third participant as an active and ever present party who takes full part in the protocol, whilst the second simply uses the third participant to provide some unrelated and random information at the start of the protocol. For this reason we shall not present an overall model of security and communication for TOPE, rather, the security and communication requirements for each of these protocols is given as required. However, we can provide a broad and informal definition that covers both TOPE protocols reviewed here.

Definition 7. *A TOPE protocol is an OPE protocol with an extra participant who does not (illegally, i.e., in secret or against the protocol) collaborate with either $\mathcal{S}$ or $\mathcal{R}$. Security is maintained as per the requirements stated in definition 5, along with the additional requirement that the extra participant cannot compute anything relating to either $f(x)$ or α.*

5.4.1 Active Third-Party TOPE

The TOPE protocol given by Chang and Lu [23] requires an active third party who takes full part in, and is present for the entire TOPE protocol. We call this third party the mediator, denoted as $\mathcal{M}$. There are three rounds of communication in this protocol, in the first round $\mathcal{R}$ sends some information to the other participants, the second round has $\mathcal{S}$ sending information and, lastly $\mathcal{M}$ sends information to $\mathcal{R}$ who then computes his evaluation. All computations are performed over a finite field $\mathbb{F}$. The full protocol is given in Figure 5.4.

Input: $\mathcal{S}$ has the polynomial $f(x) = a_0 + a_1x + \cdots + a_kx^k$ and $\mathcal{R}$ the value α.

Output: $\mathcal{R}$ receives $f(\alpha)$ and $\mathcal{S}$ gets nothing.

TOPE Protocol

1. $\mathcal{R}$ sends to $\mathcal{S}$ a set of random values, $r_1, \cdots r_k \in \mathbb{F}^k$. He also sends to $\mathcal{M}$ the values, $\alpha'_1, \cdots, \alpha'_k$ where $\alpha'_i = \alpha + r_i$ for $i = 1, \cdots, k$.

2. $\mathcal{S}$ sends to $\mathcal{R}$ a set of random values, $s_0, \cdots, s_k \in \mathbb{F}^{k+1}$. Additionally, $\mathcal{M}$ gets a set of $k+1$ values, $a'_0, \cdots, a'_k$ from $\mathcal{S}$. Where $a'_0 = a_0 + s_0 + \sum_{i=1}^{k} a_i r_i$ and $a'_i = a_i + s_i$.

3. $\mathcal{M}$ sends to $\mathcal{R}$ the value $y = a'_0 + \sum_{i=1}^{k} a' \alpha'_i$. Finally, $\mathcal{R}$ computes $f(\alpha) = y - (s_0 + \sum_{i=1}^{k} a'_i s_i)$.

Figure 5.4 : TOPE with Active Third Party [23]

The use of an actively involved third party allows for an efficient protocol, however, $\mathcal{M}$ has an integral role that is tied in with the entire protocol. As such, there may be issues with both security and availability. To elaborate, if $\mathcal{M}$ is not avaialble for the entire protocol then the OPE cannot be computed. Furthermore, the fact

that $\mathcal{M}$ is present for the entire protocol may result in some security concerns. To rectify these problems, Hanoaoka et al. [62] and Tonicelli et al. [100] developed a TOPE protocol in which the third party does not receive communications from either of the other two participants and only needs be present for the start of the protocol.

5.4.2 Commodity based TOPE

In this section we look at what is known as the commodity based TOPE given in [62, 100]. Commodity based cryptography was originally described by Beaver in [5] and involves the participants 'buying' or being assigned some (essentially random) information from a neutral third party at the start of (or before, i.e., 'offline') a given protocol. We dub this third party the initialiser, denoted as $\mathcal{I}$, and divide the TOPE protocol into two phases:

1. **Setup:** In this phase the initialiser individually assigns some correlated random information to both $\mathcal{S}$ and $\mathcal{R}$.

2. **Computation:** Here, $\mathcal{S}$ and $\mathcal{R}$ securely and privately compute an OPE using the correlated information assigned to them by $\mathcal{I}$.

As with the previously depicted OPE protocols, security is maintained if, after the protocol has been executed, $\mathcal{R}$ cannot compute any information relating to $f(x)$ (other than $f(\alpha)$) and $\mathcal{S}$ cannot compute any information relating to α. All computations are performed in the finite field $\mathbb{F}_q$, where $q > k$ is a prime number, the protocol is given in Figure 5.5.

Hanoaka et al. and Tonicelli et al. prove that their TOPE is (for the conditions that they have defined) optimal in terms of communication complexity and overall efficiency. Their scheme is elegant in its simplicity and does away with the restrictions of the active third party TOPE given in [23]. Specifically, we note that the

Input: $\mathcal{R}$ has a value α and $\mathcal{S}$ the polynomial $f(x)$ of degree at most k.

Output: $\mathcal{R}$ obtains $f(\alpha)$ and $\mathcal{S}$ gets nothing.

Setup $\mathcal{I}$ privately sends:

1. A random polynomial, $S(x)$, of degree at most k to $\mathcal{S}$.
2. A random value, d and the value $g = S(d)$ to $\mathcal{R}$.

Computation

1. $\mathcal{R}$ sends the value $l = \alpha - d$ to $\mathcal{S}$.
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial $V(x) = f(l + x) + S(x)$.
3. $\mathcal{R}$ computes $f(\alpha) = V(d) - g$.

Figure 5.5 : Commodity Based TOPE [62, 100]

setup phase of the protocol can be done at any time, even before either $\mathcal{S}$ or $\mathcal{R}$ have their respective privately held information ($f(x)$ and α). Tonicelli et al. have also rigorously proved the security of their scheme under the simulation based paradigm and their work has been used as a building block in protocols for MPC [31] and secure voting [90].

5.4.3 Extending TOPE

In this section we take the TOPE protocol shown above and extend its capabilities further modifying the underlying scheme. Our modifications are relatively simple and do not result in any dramatic changes to the efficiency of the protocol. We demonstrate three modifications to the commodity based TOPE displayed in the previous section:

1. **Multivariate Polynomial Capabilities:** Without any loss of security, we

extend the protocol to handle multivariate polynomials. Our modified scheme is just as efficient as the original scheme.

2. **Randomised Multi-Evaluation Capabilities:** By simply having $\mathcal{I}$ send extra information to $\mathcal{S}$ and $\mathcal{R}$ in the setup phase, we show how $\mathcal{R}$ can compute not only his desired evaluation ($f(\alpha)$), but also a random set of $k-1$ extra evaluations. Our modification does not add any extra communication or complexity to the computation phase.

3. **Multi-Evaluation Capabilities:** By relaxing the security constraints and slightly adapting the above modified scheme, we show how to allow $\mathcal{R}$ to compute a given set of k evaluations (that are not randomised).

TOPE With Multivariate Polynomial

Our first modification is to allow the commodity based TOPE protocol to handle multivariate polynomials. In this case $\mathcal{S}$ has the multivariate polynomial $f(x_1, \cdots, x_h)$, whilst $\mathcal{R}$ holds a range of values, $\alpha_1, \cdots, \alpha_h$ and wishes to learn the evaluation of $f(\alpha_1, \cdots, \alpha_h)$. All computations are performed in the finite field $\mathbb{F}_q$, where $q > \max(h, k)$ is a prime number. The full multivariate protocol is given in Figure 5.6.

Evaluation The security and correctness of this extended protocol is an obvious extension of the original protocol, for a full proof of this see [100]. It is easy to see that our protocol is still very efficient, only adding a multiplicative factor h onto the communication complexity of the original univariate protocol.

The probability of error for $\mathcal{R}$, i.e., the chance that $\mathcal{S}$ correctly guesses $\alpha_1, \cdots, \alpha_h$ (or $d_1, \cdots, d_h$) is given as $\frac{1}{q^h}$, as to do this $\mathcal{S}$ would have to correctly guess h values over the finite field $\mathbb{F}_q$. Although, $\mathcal{S}$ has a $\frac{1}{q}$ chance to correctly guess the evaluation value, as this requires only guessing one number. However, we note

Input: $\mathcal{R}$ has a set of values $\alpha_1, \cdots, \alpha_h$ and $\mathcal{S}$ the multivariate polynomial $f(x_1, \cdots, x_h)$ of degree at most k in any variable.

Output: $\mathcal{R}$ obtains $f(\alpha_1, \cdots, \alpha_h)$ and $\mathcal{S}$ gets nothing.

Setup $\mathcal{I}$ privately sends:

1. A random multivariate polynomial, $S(x_1, \cdots, x_h)$, of degree at most k in all variables, to $\mathcal{S}$.
2. A set of random values, $d_1, \cdots, d_h$ and the value $g = S(d_1, \cdots, d_h)$ to $\mathcal{R}$.

Computation

1. $\mathcal{R}$ sends the values $l_1, \cdots, l_h$ to $\mathcal{S}$, where $l_i = \alpha_i - d_i$ for $i = 1, \cdots, h$.
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial:

$$V(x_1, \cdots, x_h) = f(l_1, x_1, \cdots, l_h, x_h) + S(x_1, \cdots, x_h)$$

3. $\mathcal{R}$ computes:

$$f(\alpha_1, \cdots, \alpha_h) = V(d_1, \cdots, d_h) - g$$

Figure 5.6 : Commodity Based Multivariate TOPE

that security/privacy for $\mathcal{R}$ is reliant on $\mathcal{S}$ not being able to correctly compute $\mathcal{R}$'s privately held values, $\alpha_1, \cdots, \alpha_h$. Thus the probability of error here is $\frac{1}{qh}$.

We note, however, that if this value is lowered to $\frac{1}{q}$, then we can achieve even greater efficiency. To do this, we simply have $\mathcal{I}$ send one d value to $\mathcal{R}$ in the setup phase of the protocol. In the computation phase of the protocol $\mathcal{R}$ utilises the same d value as a mask for all of his $\alpha_1, \cdots, \alpha_h$ values. This modification results in a more communication efficient protocol, at the cost security. The full extension to the multivariate protocol is given in Figure 5.7.

Input: $\mathcal{R}$ has a set of values $\alpha_1, \cdots, \alpha_h$ and $\mathcal{S}$ the multivariate polynomial $f(x_1, \cdots, x_h)$ of degree at most k in any variable.

Output: $\mathcal{R}$ obtains $f(\alpha_1, \cdots, \alpha_h)$ and $\mathcal{S}$ gets nothing.

Setup $\mathcal{I}$ privately sends:

1. A random multivariate polynomial, $S(x_1, \cdots, x_h)$, of degree at most k in all variables, to $\mathcal{S}$.
2. The values d and $g = S(d, \cdots, d)$ to $\mathcal{R}$.

Computation

1. $\mathcal{R}$ sends the values $l_1, \cdots, l_h$ to $\mathcal{S}$, where $l_i = \alpha_i - d$ for $i = 1, \cdots, h$.
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial:

$$V(x_1, \cdots, x_h) = f(l_1, x_1, \cdots, l_h, x_h) + S(x_1, \cdots, x_h)$$

3. $\mathcal{R}$ computes:

$$f(\alpha_1, \cdots, \alpha_h) = V(d, \cdots, d) - g$$

Figure 5.7 : A More Efficient Commodity Based Multivariate TOPE

As discussed, the probability of error here is only $\frac{1}{q}$ as all $\mathcal{S}$ has to do is correctly guess $d \in \mathbb{F}_q$ to easily compute all $\alpha_1, \cdots, \alpha_h$ values.

TOPE With Randomised Multi-Evaluation

Our next modification allows $\mathcal{R}$ to compute not only his evaluation $f(\alpha)$, but also a set of $k-1$ extra (random) evaluations as well.

This modification is efficient and only effects the setup phase, allowing for a computation phase that is just as efficient as the original unmodified protocol. The benefit of this is evident when we consider that often-times in protocols such as

MPC, a lot of computation is delegated to the offline or setup phase in order to make the actual computation phase (or 'online' phase) as efficient as possible. The logic behind this choice being that the setup phase can be done at any time, well in advance of when the actual computation is needed.

In order to compute the extra, random, evaluations we have $\mathcal{I}$ send to $\mathcal{R}$ a set of extra values $d_1, \cdots, d_{k-1}$ during the setup phase. The computation phase then proceeds (from a communication point of view) exactly as before, with the only change being some small, extra, computations performed privately by $\mathcal{R}$. The full protocol is given in Figure 5.8.

The main benefit of our protocol is that we are able to compute k evaluations for the same communication complexity (in the computation phase) as the original protocol. A naive approach to this would result in a multiplicative increase of k, something our protocol manages to avoid by simply designating all extra communication to the setup phase.

Evaluation From $\mathcal{S}$'s point of view the actual protocol is unchanged from the original protocol. Thus it only remains to show that $\mathcal{R}$ cannot compute anything extra from the multiple evaluation points he has received.

Theorem 11. *The randomised multi-evaluation OPE protocol maintains privacy for $\mathcal{S}$.*

Proof. The proof of this is quite simple and is a result of Shamir's secret sharing scheme [97].

At the end of the modified OPE protocol $\mathcal{R}$ will have obtained k evaluations of a k degree polynomial: $f(x) = a_0 + a_1 x + \cdots + a_k x^k$. as such, $\mathcal{R}$ holds a system composed of $k+1$ unknowns and k independent equations. In other words, $\mathcal{R}$ has k shares of a Shamir polynomial of degree k. As per Shamir's scheme, it is known that

Input: $\mathcal{R}$ has the value α and $\mathcal{S}$ the polynomial $f(x)$ of degree k or more

Output: $\mathcal{R}$ obtains $f(\alpha)$ as well as $k-1$ random evaluations of $\mathcal{S}$'s polynomial of the form: $f(\beta_1), \cdots, f(\beta_{k-1})$. As before, $\mathcal{S}$ gets nothing.

Setup $\mathcal{I}$ privately sends:

1. A random polynomial, $S(x)$, of degree k or more to $\mathcal{S}$.
2. A set of random values, $d_0, \cdots, d_{k-1}$, and the values $g_i = S(d_i)$ to $\mathcal{R}$, for $i = 0, \cdots, k-1$.

Computation

1. $\mathcal{R}$ computes and sends to $\mathcal{S}$ the value $l = \alpha - d_0$. Privately, $\mathcal{R}$ also computes $k-1$ values, of the form $\beta_i = l + d_i$, for $i = 1, \cdots, k-1$
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial $V(x) = f(l+x) + S(x)$.
3. $\mathcal{R}$ computes his evaluation as: $f(\alpha) = V(d_0) - g_0$. The $k-1$ random evaluations are computed in much the same fashion: $f(\beta_i) = V(d_i) - g_i$.

Figure 5.8 : Commodity Based Randomised Multi-Evaluation OPE

$k+1$ shares are needed to compute the polynomial. Therefore, $\mathcal{R}$ cannot compute anything extra about $\mathcal{S}$'s polynomial, other than what his evaluation points already tell him. □

The benefits of this protocol are evident when looking at the uses to which the original protocol was put to. Specifically, the original OPE is used as a multiplication protocol in MPC [31] and is also the backbone of a secure voting protocol [90]. Our modifications would allow for increases in efficiency, for both of these purposes i.e., computing multiple multiplications simultaneously and/or evaluating multiple votes simultaneously in each respective protocol.

TOPE With Multi-Evaluation

In this final extension to the commodity based TOPE protocol we show that by lessening the original security requirements of the protocol, we can modify the previously discussed randomised multi-evaluation scheme to allow $\mathcal{R}$ to actually choose all evaluation points. To do this we modify the setup phase and allow $\mathcal{R}$ to communicate with $\mathcal{I}$. This modification does not actually lessen security in any fashion, as $\mathcal{I}$ will still not be able to learn anything relating to either $\mathcal{S}$'s or $\mathcal{R}$'s private information. Furthermore, this new protocol has the same communication complexity as the randomised multi-evaluation protocol and we still do not require $\mathcal{I}$ to be an active participant throughout the protocol (as in [23]) i.e., $\mathcal{I}$ is only present for the setup phase.

To summarise our modification, we take the randomised multi-evaluation protocol and add an extra level of communication in the setup phase, whereby $\mathcal{R}$ sends to $\mathcal{I}$ a specific set of $d_1, \cdots, d_k$ points to be evaluated by $S(x)$. This allows $\mathcal{R}$ to choose the evaluation points he requires, as opposed to them being random. The full protocol is given in Figure 5.9.

As with the randomised protocol, the benefits of this scheme have applications in both MPC and privacy preserving protocols. However, with this scheme there is no need to use randomised evaluations, rather a set of predetermined evaluations can be chosen, allowing for a for more useful and versatile protocol.

Evaluation From a security perspective this protocol is (for $\mathcal{S}$ and $\mathcal{R}$) exactly the same as the randomised protocol, in that neither $\mathcal{R}$ nor $\mathcal{S}$ can compute any information they do not get directly from the protocol or already have. As the protocol is exactly the same in terms of what is shared between $\mathcal{S}$ and $\mathcal{R}$ we can rest assured that security is maintained for this case. When looking at the extra

Input: $\mathcal{R}$ has the values $\alpha_1, \cdots, \alpha_k$ and $\mathcal{S}$ the polynomial $f(x)$ of degree k or more

Output: $\mathcal{R}$ obtains $f(\alpha_1), \cdots, f(\alpha_k)$. As before, $\mathcal{S}$ gets nothing.

Setup

1. $\mathcal{I}$ sends a random polynomial, $S(x)$, of degree k or more to $\mathcal{S}$.
2. $\mathcal{R}$ computes and sends to $\mathcal{I}$ the values $d_1, \cdots, d_k$. Where $d_i = l - \alpha_i$ for $i = 1, \cdots k$ and l is a random, private value chosen by $\mathcal{R}$.
3. $\mathcal{I}$ sends to $\mathcal{R}$ the values $g_1, \cdots, g_k$ where $g_i = S(d_i)$.

Computation

1. $\mathcal{R}$ sends to $\mathcal{S}$ the value l.
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial $V(x) = f(l - x) + S(x)$.
3. $\mathcal{R}$ computes his k evaluations as: $f(\alpha_i) = V(d_i) - g_i$ for $i = 1, \cdots, k$

Figure 5.9 : Commodity Based TOPE with Multi-Evaluation Capabilities

information given to $\mathcal{I}$ by $\mathcal{R}$ it is easy to see that $\mathcal{I}$ cannot compute anything relating to any of $\mathcal{R}$'s evaluation points. This is because, each of the d_i values is essentially random from $\mathcal{I}$'s point of view, to correctly guess any α_i the initialiser would have to guess $l \in \mathbb{F}_q$. This gives $\mathcal{I}$ a $\frac{1}{q}$ chance of correctly computing any extra information, the same probability as in the original scheme given by Tonicelli et al. [100], and the same probability that $\mathcal{S}$ has of correctly guessing the evaluation points.

We note that performing k OPEs would result in an overall probability of $\frac{1}{qk}$, however this comes at a far greater cost to communication.

5.5 Flaws in Bo et al. OPE Scheme

In this section we show that the OPE protocol devised by Bo et al. [15] is not secure. The essential premise of this scheme (as per the author's claims) is that it achieves unconditional security with only two participants, $\mathcal{R}$ and $\mathcal{S}$. However, as we mentioned at the start of this Chapter, it has long been established that a two party unconditionally secure protocol is impossible [28, 38]. To demonstrate this fact we display Bo et al.'s protocol in Figure 5.10 and then discuss the flaws in their proposed OPE.

The idea behind this protocol is to utilise a series of random values as masks, in order to preserve the privacy of both $\mathcal{S}$ and $\mathcal{R}$. However, after learning the evaluation $(f(\alpha))$ the receiver, $\mathcal{R}$, is able to go back and actually compute the masks used by $\mathcal{S}$. This then allows $\mathcal{R}$ to break the protocol by computing $f(x)$. The exact method by which this is possible is given below in an example in which we set $k = 1$ (the degree of $\mathcal{S}$'s polynomial).

As stated, $k = 1$, so $l = 1$ as well. Assume that the protocol has been completed and $\mathcal{R}$ has computed $f(\alpha)$. We now draw the reader's attention to step 3 of the protocol, in which $\mathcal{R}$ is assigned the following pieces of information:

1. $D_0 = Ha_0$.
2. $D_1 = Ha_1\Delta_{1_1}$ (as $a_2 = 0$).

where H, a_0 and a_1 are unknowns. Now, at the end of the protocol $\mathcal{R}$ also has the equation $f(\alpha) = a_0 + \alpha a_1$, as before a_0 and a_1 are unknowns. Combining these pieces of information gives the following system of equations:

$$D_0 = Ha_0$$

$$D_1 = Ha_1\Delta_{1_1}$$

$$f(\alpha) = a_0 + a_1\alpha$$

Input: $\mathcal{R}$ has the values α and $\mathcal{S}$ the polynomial $f(x)$ of degree k or more

Output: $\mathcal{R}$ obtains $f(\alpha)$ and $\mathcal{S}$ gets nothing.

Preliminaries: Let $l = \lfloor \frac{k}{2} \rfloor + 1$ if k is odd and $l = \frac{k}{2}$ if k is even. All values are drawn from the field $\mathbb{Z}_q \setminus \{0\}$ where q is a large prime.

OPE Protocol

1. $\mathcal{R}$ privately selects the random values β_1, β_2, T_1, T_2 and $r_1, \cdots, r_l$. He uses these values to compute l values of the form $r'_1, \cdots, r'_l$, such that $r'_j = T_2^{-1} r_j$ for $j = 1, \cdots, l$.

2. $\mathcal{R}$ then sends to $\mathcal{S}$ the values $\Delta_j = (\Delta_{j_1}, \Delta_{j_2})$ where:

$$\Delta_{j_1} = T_1 r_j \alpha^{2j-1} + \beta_1 r'_j$$

$$\Delta_{j_2} = T_1 r_j \alpha^{2j} + \beta_2 r'_j$$

3. $\mathcal{S}$ privately computes the random value H and sends to $\mathcal{R}$ the values $D_0, \cdots, D_l$ where $D_0 = Ha_0$ and $D_j = Ha_{2j-1}\Delta_{j_1} + Ha_{2j}\Delta_{j_2}$, where $j = 1, \cdots l$ and $a_{2l} = 0$ if k is odd.

4. $\mathcal{R}$ computes $M = T_1 T_2 D_0 + \sum_{j=1}^{l} D_j (r'_j)^{-1}$ and then sends to $\mathcal{S}$ the value $M_1 = M\beta_1^{-1}$

5. $\mathcal{S}$ sends to $\mathcal{R}$ the value $S_1 = M_1 - H \sum_{j=1}^{l} a_{2j-1}$.

6. Using this value, $\mathcal{R}$ sends to $\mathcal{S}$ the value $M_2 = S_1 \beta_1 \beta_2^{-1}$.

7. Following this, $\mathcal{S}$ sends to $\mathcal{R}$ the value $S_2 = \left(M_2 - H \sum_{j=1}^{l} a_{2j} \right) H^{-1}$

8. Finally, $\mathcal{R}$ computes $f(\alpha) = S_2 \beta_2 T_1^{-1} T_2^{-1}$

Figure 5.10 : Flawed OPE protocol [15]

To solve this system, we multiply the third equation by H and substitute the first two equations into this new third equation. Doing so gives, $f(\alpha)H = D_0 + \alpha D_1$. Solving this gives the value of $H = (D_0 + \alpha D_1)(f(\alpha))^{-1}$. Now that H is known we can compute a_0 and a_1 with ease:

$$a_0 = H^{-1}D_0$$

$$a_1 = H^{-1}\Delta_{1_1}^{-1}D_1$$

As a result of this $\mathcal{R}$ is able to compute the entirety of $\mathcal{S}$'s polynomial $f(x)$, thereby resulting in a flawed scheme that does not ensure either security or privacy. We note that our attack will also work with all possible cases, however $k = 1$ was used in order to easily demonstrate the flaws in this protocol.

5.6 Conclusion

In this Chapter we critically reviewed the exiting information theoretic OPE protocols. Additionally we made several key contributions to the field of information theoretic OPE:

1. We adapted a DOT protocol into a flexible DOPE protocol that is equivalent to an existing DOPE protocol presented in [75], displaying the strong link between the relatively new field of DOPE and the existing and well researched field of DOT.

2. We created 3 extensions of a well known TOPE protocol, resulting in the following improvements:

   (a) Multivariate capabilities.

   (b) Randomised multi-evaluation capabilities.

   (c) Multi-evaluation capabilities with relaxed security.

3. Demonstration of the flaws inherent in the OPE given by Bo et al. [15].

OPE is a relatively new field which is continuing to grow, with new results and applications appearing frequently. As such, there is still many more opportunities for research within this field, particularly in terms of the myriad of applications to which this protocol can be put towards. We demonstrate this in the next two Chapters, by constructing a robust MPC protocol on the back of the commodity based TOPE protocol reviewed here.

Chapter 6

Efficient Information Theoretic Multi-Party Computation from Oblivious Linear Evaluation

6.1 Introduction

In terms of efficiency and communication complexity, multiplication in MPC has always been a large bottleneck. The typical method employed by most current protocols has been to utilise Beaver's method [4], which relies on some precomputed information. In this Chapter we introduce an OLE-based MPC protocol which also relies on some precomputed information.

For a specific family of functions which consist of a sum of monomials, our proposed protocol has a more efficient communication complexity than Beaver's protocol by a multiplicative factor of t. Furthermore, to compute a share to a multiplication, a participant in our protocol need only communicate with one other participant; unlike Beaver's protocol which requires a participant to contact at least t other participants.

This result is achieved by utilising a special case of OPE, wherein $f(x)$ is of degree at most one, known as oblivious linear evaluation (OLE). Specifically, we utilise OLE for the purpose of performing multiplication in multi-party computation (MPC).

To refresh the reader, MPC allows a set of n mutually distrustful participants to compute any given function across their private inputs, without revealing any information relating to their private inputs. We focus on the threshold setting, where an MPC protocol is considered secure if a set of t or less participants, where $t < n$, cannot gain any information relating to another participant's private input,

other than what the output of the protocol gives them. More formally:

Definition 8. *A (t, n) threshold MPC protocol allows a set of n participants, $P_1, \cdots, P_n$ with respective private inputs, $x_1, \cdots, x_n$ to compute a given function, $f(x_1, \cdots, x_n)$.*

Privacy is maintained if, after completion of the protocol, an adversary controlling any subset of up to t participants ($t < n$), cannot learn more information (about other participant's private inputs) than what could be derived from each participant's individual, private input and the output of the protocol.

Traditionally, the adversary is classified as either passive or malicious. Participants under control of a passive adversary may share information with one another but do not deviate from the MPC protocol. Participants under control of a malicious adversary also share information but may act arbitrarily, i.e., they do not necessarily follow the protocol. Another aspect of the adversary considered in an MPC protocol is the resources it has at its command. Specifically, an unconditionally (information theoretic) secure MPC protocol is secure against a computationally unbounded adversary. Whilst a conditionally (computationally) secure MPC protocol is secure against a computationally bounded adversary.

This Chapter focuses on building a (t, n) threshold MPC scheme, secure against a passive (semi-honest) adversary. We show the construction of an efficient MPC scheme based on OLE. In the next section we give some background and motivation on this topic, following this we then discuss our contribution in depth.

6.1.1 Background

MPC is a powerful tool that can be used to solve practically any given problem involving a set of distrustful parties. In classical, unconditionally secure protocols [7, 24, 94] each participant, P_i $(i = 1, \cdots, n)$ shares their private input, x_i by utilising Shamir's secret sharing scheme [97] to distribute shares to all participants.

To compute a given function, $f(x_1, \cdots, x_n)$, participants need simply perform all computations on the shares of each input value. For instance, if a participant wants to compute a share relating to the sum of two distributed input values he simply adds his two corresponding shares together. At the end of the protocol, a set of $t+1$ or more participants then pool their information to reconstruct the output.

Due to the (+, +)-homomorphic nature of Shamir's scheme [8] participants can easily compute any linear operation by privately computing on their shares. However, since the inception of MPC [60] the largest limiting factor has been the high amount of resources required to compute a multiplication. Specifically, multiplying the shares associated with two secrets does actually result in shares of a polynomial with the correct free term (i.e., the desired multiplication). However, the issue is that the resulting polynomial will be of degree $2t$, thereby making it impossible for $t+1$ participants to reconstruct the results of the multiplication. As a direct result of this, the very first MPC protocols [7] tended to rely upon complex, multi-round protocols that essentially reduces the degree of the multiplication polynomial.

Perhaps the most widely known and efficient method of computing a multiplication in an MPC protocol is known as Beaver's method (A.K.A Beaver's triples) [4]. For completeness we review this protocol below.

Beaver's Method

Beaver's method [4] for computing a multiplication in MPC relies on some pre-shared information known as a triple. Specifically, a triple is composed of three values, a, b, and c where $a \cdot b = c$ and $a, b, c \in \mathbb{F}_q$ such that $q > n$ and q is a prime number. Each participant has a share of these triples, such that participant P_k $(k = 1, \cdots, n)$ receives the shares a_k, b_k and c_k relating to (respectively) a, b, and c.

Suppose we have participants P_i with input x_i and P_j with input x_j for $i, j =$

$1, \cdots, n$ and $i \neq j$. To compute shares of the multiplication $\gamma = x_i \cdot x_j$ we first have both P_i and P_j distribute shares of their private values among the other participants, where P_k gets x_{i_k} relating to x_i and x_{j_k} relating to x_j. To compute a share, γ_k relating to the product γ, a set of at least $t+1$ participants execute the following steps:

1. Each participant, P_k, computes $z_k = x_{i_k} - a_k$ and $v_k = x_{j_k} - b_k$, where z_k is a share of the value $z = x_i - a$ and v_k is a share of $v = x_j - b$.

2. A set of at least $t+1$ participants broadcast their shares, z_k and v_k amongst themselves.

3. Participants publicly reconstruct the values of z and v using the shares z_k and v_k, respectively.

4. P_k computes his share of γ as $\gamma_k = zv + zb_k + va_k + c_k$.

5. $t+1$ or more participants can reconstruct $\gamma = x_i \cdot x_j$ by pooling their shares.

In order to construct z and v a set of $t+1$ participants is required to cooperate. If all participants in this set wish to compute these values (and consequently, the multiplication) then each participant must both receive and send t messages. Since each message would consist of a constant amount of elements from the field $\mathbb{F}_q$ (i.e., the values z_k and v_k) the communication complexity of this protocol can be given as $\mathcal{O}(t^2 \log q)$.

Many recent MPC protocols utilise a resource intensive computationally secure offline phase to compute these multiplication triples. The actual MPC is then carried out in a faster information theoretic online phase. For our purposes, we focus solely on the information theoretic online phase. It suffices to assume that participants gain the shares of the triples via an external party known as an initialiser, who (after computing and distributing the shares of the triples) does not take part in

the actual MPC protocol. In the next section we review the OLE based two-party protocol given by Döttling et al. [53].

TinyOLE

Recently Döttling et al. [53] proposed a two-party protocol ($n = 2$) in which the two participants, P_1 and P_2 use OLE to compute shares to a multiplication. Specifically, they use OLE to compute multiplication triples in an offline phase. Their scheme utilises a simple additive secret sharing scheme wherein a given value, a (for example), is represented as $a = a_1 + a_2$, across a finite field $\mathbb{F}$; where P_2 has the share a_2 and P_1 gets a_1. Addition in their scheme consists of simply adding shares together. Multiplication is achieved by utilising OLE in a black-box fashion.

To compute a multiplication of two distributed (and not necessarily known) values, a and b, they rely on the fact that: $ab = a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2$. To compute the "troublesome" terms of the form $c = a_1b_2$ they utilise a black-box OLE. Essentially, P_1 acts as a sender and submits the polynomial $f(x) = a_1x - c_1$ where c_1 is a randomly chosen value. The second participant, P_2 acts as receiver and submits $\alpha = b_2$. Both participants send their values to a black-box OLE, with P_2 receiving back $f(\alpha) = a_1b_2 - c_1$. If we set $c_2 = f(\alpha)$ then each participant now holds a share of c as $c = c_1 + c_2$. To compute shares to the entire multiplication it is easy to see that at least 2 OLEs are needed.

Döttling et al. specifically use this method in a computationally secure offline phase to compute random multiplication triples, where the values of a and b are not actually known to either participant. Our proposed scheme differs to theirs in that we wish to utilise OLE in an information theoretic, online phase to compute the multiplication of known input values for a given MPC function.

6.1.2 Our Contribution

In this section we summarise our proposed MPC scheme which utilises OLE to compute shares to a given multiplication. In contrast to the methods discussed above our protocol obtains the following desirable properties:

1. Unlike Beaver's scheme [4] our proposed protocol only requires communication between two participants to compute a given share to a multiplication i.e., a participant may compute his share without the assistance of t other participants. We achieve this result by having one designated participant who acts as a sender in an OLE. The other participants need simply privately compute an OLE with this sender participant to compute a share to a multiplication. As a result of this, the communication complexity of our protocol is $\mathcal{O}(t \log q)$, which is more efficient than Beaver's (at $\mathcal{O}(t^2 \log q)$) by a multiplicative factor of t.

2. We do not rely on a black-box method of OLE and instead provide a specific construction. Our OLE multiplication scheme is based on the information theoretic protocol given in [62, 100]. This scheme, like Beaver's scheme, relies on some precomputed information which can be produced via an offline phase or an initialiser. Since we wish to focus solely on the information theoretic OLE-based MPC scheme itself we will assume that the information is provided via an initialiser.

3. Our scheme only utilises one OLE per participant to compute a multiplication. In a two party protocol we would only need one OLE. So, for an individual participant to compute his share (in either a multi-party or two-party protocol) the complexity cost is just $\mathcal{O}(\log q)$.

4. Lastly, unlike the TinyOLE scheme [53], our scheme is scalable, in that it

extends to the multi-party case with n participants. In fact, computing n shares (one for each participant) to a single multiplication requires only $n-1$ OLEs, one for each individual participant to compute his share. We note that utilising all n participants is not actually necessary. We only really require a set of $t+1$ participants, enough to compute the output of a given multiplication at the end of the protocol.

As noted, there are restrictions to the usefulness of our protocol, namely that it is only efficient for specific types of functions which are composed of a sum of monomials. This is explored further in Chapter 7, Section 7.6.

6.1.3 Outline

The rest of the Chapter is organised as follows. In section 6.2 we go over some of the sub protocols and tools used in our proposed MPC protocol. Section 6.3 gives a high level overview of our protocol as well as a model for security. The actual construction for our protocol is given in section 6.4, along with an evaluation and proof of correctness and security.

6.2 Preliminaries

The building blocks of our protocol are Shamir's secret sharing scheme, and the previously dubbed 'commodity based TOPE' protocol given by Hanaoka et al. in [62]. Since these protocols are extensively discussed in previous Chapters we have neglected to include them again here. For reference, however, we have included a description of Hanaoka et al.'s OPE scheme in Figure 6.1.

Input: $\mathcal{R}$ has a value α and $\mathcal{S}$ the polynomial $f(x)$ of degree at most t.

Output: $\mathcal{R}$ obtains $f(\alpha)$ and S gets nothing.

Setup The initialiser privately sends:

1. A random polynomial, $S(x)$, of degree at most t to $\mathcal{S}$.
2. A random value, d and the value $g = S(d)$ to $\mathcal{R}$.

Computation

1. $\mathcal{R}$ sends the value $l = \alpha - d$ to $\mathcal{S}$.
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial $V(x) = f(l + x) + S(x)$.
3. $\mathcal{R}$ computes $f(\alpha) = V(d) - g$.

Figure 6.1 : Information Theoretic OPE [62, 100]

6.3 Model

This section presents a high level overview of our protocol and a set of criteria for evaluating the security of our scheme. We use the traditional setting of MPC protocols. That is, each party P_j $(1 \leq j \leq n)$ distributes its private input, x_j amongst all participants, using a Shamir (t, n) threshold scheme. Linear functions can be computed by each participant privately. In order to perform multiplication, however, we must utilise OLE.

6.3.1 Overview

Suppose we have a set of n participants who wish to compute shares to the value $\gamma = x_i \cdot x_j$, where x_i and x_j are the respective private input values of participants P_i and P_j, for $1 \leq i, j \leq n$ and $i \neq j$. Further suppose, that P_j utilises the polynomial $f_j(x)$ to share x_j among all participants (via Shamir's secret sharing scheme), such

that a given participant P_k $(k = 1, \cdots, n)$ receives the share $x_{j_k} = f_j(k)$ of x_j.

A simple method for computing shares of γ is to have each participant, P_k, simply send his share, x_{j_k}, to P_i who can then send back the value $\gamma_k = x_i x_{j_k}$. Due to the (+, +)-homomorphic nature of Shamir's secret sharing scheme the value γ_k is a share corresponding to the polynomial $\Gamma(x) = x_i f_j(x)$ with free term $x_i \cdot x_j$. The obvious problem with this simple protocol is that neither P_i's nor P_j's privacy is maintained.

To keep P_i's input, x_i, private we can have P_i introduce a random, private masking polynomial, $h_i(x)$, of degree at most t, with free term $h_i(0) = 0$. Now, when he receives a given share, x_{j_k}, we require P_i to send back $\gamma_k = x_i x_{j_k} + h_i(k)$. Each P_k now holds shares to the polynomial $\Gamma(x) = x_i f_j(x) + h_i(x)$. Intrinsically we can see that, due to Shamir's secret sharing scheme, the protocol is now t-private with respect to P_i, as a set of t participants with t shares cannot compute any information relating to the effectively random polynomial $\Gamma(x)$. It remains to ensure the privacy of P_j.

Surprisingly, ensuring that P_j's privacy is maintained is actually quite simple. Rather than having each P_k simply hand his share to P_i we instead have P_k and P_i utilise an OLE protocol, where P_k acts as the receiver and P_i as the sender. First P_i computes two polynomials, $f_i(x) = x_i \cdot x$ and $h_i(x)$ (the masking polynomial, as before). Each P_k $(1 \leq k \leq n)$ then acts as the receiver and executes an OPE protocol with P_i (who acts as the sender) to privately evaluate P_i's polynomial, $f_i(x)$ at the point x_{j_k}, as before P_i adds the masking polynomial to his computation.

Since the OLE protocol does not allow P_i to learn the evaluation point then the protocol can now be considered t-private for both P_i and P_j. Specifically, P_i's privacy is maintained via the masking polynomial and P_j's privacy is maintained via Shamir's secret sharing scheme and the OLE protocol. An overview of this is

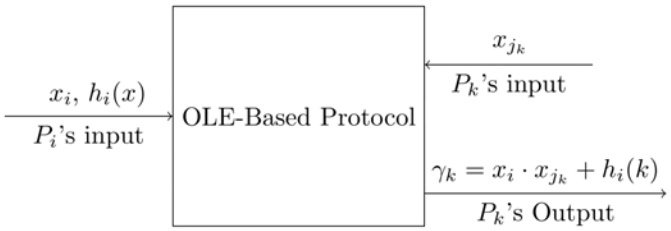

Figure 6.2 : Overview of the protocol

given in Figure 6.2. Note that P_i will also use his share from x_j and compute his own share of $\gamma = x_i \cdot x_j$ (of course, there is no need to perform OLE, as he plays the role of the sender and receiver at the same time).

6.3.2 Security and Correctness

In order to prove the security and correctness of our proposed scheme we will evaluate it against the following criteria specified below:

1. **Correctness** – Upon completion of the protocol each participant, P_k holds a share, γ_k of the polynomial $\Gamma(x)$, of degree at most t with free term $\Gamma(0) = x_i \cdot x_j$.

2. **Privacy** – A set of t or less participants, not including either P_i or P_j, cannot reduce their uncertainty of x_i or x_j.

3. **Privacy with respect to** P_i – A set of t or less participants, including P_j, cannot reduce their uncertainty of x_i.

4. **Privacy with respect to** P_j – A set of t or less participants, including P_i, cannot reduce their uncertainty of x_j.

We note that the last three criterion presented here simply encapsulate the notion of privacy given in definition 8.

6.4 Proposed OLE-Based MPC Protocol

Similar to the OPE protocol given in Figure 6.1, our proposed multiplication protocol consists of two phases:

1. **The Setup Phase:** Where the initialiser privately sends some (essentially random) information to each participant involved in the protocol.

2. **The Computation Phase:** Where participants are able to compute shares to the multiplication.

Where our scheme differs to the original OPE, however, is that we utilise a new masking polynomial ($h_i(x)$) and we limit the degree of the receivers polynomial to a maximum value of 1 (i.e., an OLE scheme instead of an OPE scheme).

As per section 6.3.1 suppose we have a set of n participants $P_1, \cdots, P_n$, with respective private inputs $x_1, \cdots, x_n$, who wish to compute shares of the value $\gamma = x_i \times x_j$ where $i, j \in [1, n]$ and $i \neq j$. Participant P_j first privately distributes shares for x_j amongst all participants, using the polynomial $f_j(x)$, such that P_k $(1 \leq k \leq n)$ gets the share $x_{j_k} = f_j(k)$. To compute a share γ_k, of γ each P_k cooperates with P_i to execute our modified OLE protocol, with P_k essentially acting as the receiver and P_i acting as the sender for each P_k. Note that all computations are performed in the field $\mathbb{F}_q$ where q is a prime number such that $q > n$. The full protocol is given in the *OLEMult* protocol presented in Figure 6.3. We note that the exchange between P_k and P_i can actually just be viewed as one OLE if we set $f_{i,k}(x) = x_i \cdot x + h_i(k)$, it is only for clarity of purpose that we delineate $h(x)$ as a masking polynomial.

In order to compute a share, P_k and P_i exchange exactly 3 field elements (l and $V(x)$). This gives a communication complexity of $\mathcal{O}(\log q)$. Therefore, the overall communication complexity, required for all of the n participants to compute his share can be given as $\mathcal{O}(n \log q)$. However, since the protocol is based on Shamir's

Input: P_i has x_i and P_k has the share x_{j_k}, of x_j. **Output:** P_k obtains the share γ_k, of $\Gamma(x)$ where $\Gamma(0) = x_i \cdot x_j$.

Setup The initialiser privately sends:

1. A set of $n-1$ random polynomials, $S_{i_k}(x)$, of degree at most 1 to P_i where $k = 1, \cdots, n$ and $k \neq i$.
2. A random value, d_k and the value $g_k = S_{i_k}(d_k)$ to every participant P_k.

Computation P_i privately computes:

- A masking polynomial, $h_i(x)$ of degree at most t with $h_i(0) = 0$.
- The multiplication polynomial, $f_i(x) = x_i \cdot x$.

Each P_k then privately executes the following steps with P_i:

1. P_k sends the value $l_k = x_{j_k} - d_k$ to P_i.
2. P_i then computes and sends to P_k the polynomial $V_k(x)$ which is computed as:

$$V_k(x) = h_i(k) + f_i(x + l_k) + S_{i_k}(x)$$

3. P_k computes his share of γ as $\gamma_k = V_k(d_k) - g_k$.

Figure 6.3 : *OLEMult* Protocol - An information theoretic, OLE-based multiplication protocol for MPC

(t, n) secret sharing scheme [97] we actually only require $t+1$ participants to ensure the output can be constructed. This gives a communication complexity of $\mathcal{O}(t \log q)$.

6.4.1 Evaluation

In this section we evaluate the proposed protocol against the set of security criteria given in section 6.3.2. We note that all four of these criterion evaluate the specific multiplication protocol and not the actual MPC itself. That is, we evaluate the multiplication protocol only and assume that participants have not yet reconstructed the actual output of the MPC.

Correctness At the end of our protocol each participant, P_k, will now have a share of the polynomial: $\Gamma(x) = h_i(x) + x_i f_j(x)$. Since the free term of $h_i(x)$ is equal to zero, we can say that $\Gamma(0) = x_i f_j(0)$. Now, $f_j(x)$ has the free term x_j and both $h_i(x)$ and $f_j(x)$ are of degree at most t. As a result of this we can conclude that correctness is achieved, as each P_k has a share to a polynomial, $\Gamma(x)$, of degree at most t, with free term equal to $x_i \cdot x_j$.

Privacy

Theorem 12. *A set of t participants, not including P_i or P_j, cannot compute any information relating to x_i or x_j.*

In order to prove this we must first show that the modified OLE protocol is secure. Following this, we need to prove that a set of t shares relating to the multiplication reveals no information.

Proof. Suppose that a given participant, P_k executes the multiplication protocol with P_i. After sending l_k to P_i he receivers the polynomial $V_k(x) = h_i(k) + f_i(x + l_k) + S_{i_k}(x)$ which we can simplify as $V_k(x) = v_k + z_k x$. Let $S_{i_k}(x) = \kappa_k + \omega_k x$ and recall that $f_i(x) = x_i \cdot x$, then we can rewrite the equation as $V(x) = h_i(k) + x_i l_k + \kappa_k(x_i + \omega_k)x$. This gives P_k the following information:

$$v_k = h_i(k) + x_i l_k + \kappa_k$$

$$z_k = x_i + \omega_k$$

Since the values ω_k and κ_k (as well as the coefficients of $h_i(x)$) are chosen at random, P_k cannot gain any information from the above equations. The next step in the protocol is for P_k to compute $\gamma_k = V(d_k) - g_k$, which can be written as $\gamma_k = h_i(k) + x_i x_{i_j}$. Individually, this gives no information to P_k as he does not know the value of either $h_i(k)$ or x_i, it remains to be seen if a coalition of participants can compute any information.

Without loss of generality suppose that the first set of t participants, $P_1, \cdots, P_t$ pool their information together. Let $h_i(x) = m_1 x + m_2 x^2 + \cdots + m_t x^t$, then the coalition can compute the following system:

$$\begin{aligned}
\gamma_1 &= x_i \cdot x_{i_1} + m_1 + m_2 + \cdots + m_t \\
\gamma_2 &= x_i \cdot x_{i_2} + 2m_1 + 4m_2 + \cdots + 2^t m_t \\
&\vdots \\
\gamma_t &= x_i \cdot x_{i_t} + tm_1 + t^2 m_2 + \cdots + t^t m_t
\end{aligned}$$

Due to the perfectness of of Shamir's secret sharing scheme [36, 97] the above system does not reveal any information to the participants as they effectively have a set of t shares relating to a degree t polynomial. This becomes even more evident when we take into account that $x_{i_k} = f_j(k)$ meaning that each P_k has a share of the polynomial $\Gamma = x_i f_j(x) + h_i(x)$.

The end result being that a coalition of t participants cannot reduce their uncertainty of x_i. The same is also true for x_j, as collectively the coalition only has t shares of $f_j(x)$. □

Privacy with respect to P_i

Theorem 13. *A set of t participants, including P_j, cannot compute any information relating to x_i.*

Proof. The proof of this is similar to the proof of Theorem 12 along with some extra information. Namely, we now assume that the coalition of participants has the values of both $f_j(x)$ and, consequently x_j. The first, obvious ramification of this is that the coalition now know the shares of every other participant relating to x_j. This actually gives them no advantage, in regards to the OLE, as they do not know

(and cannot compute) the values given to the other participants by the initialiser (namely d_k and g_k). We therefore only need to prove that knowing $f_j(x)$ reveals no information relating to x_i.

As before, at the end of the protocol each participant has a share to the polynomial $\Gamma(x) = x_i f_j(x) + h_i(x)$. It is easy to see that if the coalition can compute $\Gamma(x)$ or even $h_i(x)$ then they can easily compute x_i. However, the coalition do not hold direct shares to $h_i(x)$, so even knowing $h_i(0) = 0$ gives them nothing. Furthermore, to compute any information relating to $\Gamma(x)$ would require the coalition to compute a solution to the system given in the proof of Theorem 12.

Computing a solution to this system is analogous to solving a system of equations composed of $t + 1$ unknowns (x_i and the coefficients of $h_i(x)$) and t equations. We can therefore conclude that a set of participants, including P_j cannot reduce their uncertainty of x_i. □

Privacy with respect to P_j

Theorem 14. *A set of t participants, including P_i, cannot compute any information relating to x_j.*

Proof. In the proof of Theorem 12 it was shown that the modified OLE is secure, therefore to prove the above Theorem we need to show that a coalition of t participants, including P_i, with t shares relating to $\Gamma(x) = x_i f_j(x) + h_i(x)$ and t shares of $f_j(x)$ cannot compute any information relating to x_j. First, let $f_j(x) = x_j + W_1 x + \cdots + W_t x^t$ and assume, as before, that a coalition composed of the first t participants (which includes P_i) pool their knowledge. They can construct the following system from their shares of $\Gamma(x)$:

$$\gamma_1 = x_i \cdot (x_j + W_1 + \cdots + W_t) + h_i(1)$$
$$\gamma_2 = x_i \cdot (x_j + 2W_1 + \cdots + 2^t W_t) + h_i(2)$$
$$\vdots$$
$$\gamma_t = x_i \cdot (x_j + tW_1 + \cdots + t^t W_t) + h_i(t)$$

From the shares of $f_j(x)$ we get:

$$x_{j_1} = x_j + W_1 + \cdots + W_t$$
$$x_{j_2} = x_j + 2W_1 + \cdots + 2^t W_t$$
$$\vdots$$
$$x_{j_t} = x_j + tW_1 + \cdots + t^t W_t$$

It is easy to see that the two systems are actually linearly dependent. Since the values of x_i and $h_i(x)$ are known to the coalition, this results in a system composed of $t+1$ unknowns (the coefficients of $f_j(x)$) and only t linearly independent equations. The net result of this is that each value of x_j is, from the point of view of the coalition, equally likely. Meaning that they cannot compute any information relating to x_j. $\square$

Chapter 7

OLE-Based MPC Secure Against a Malicious Adversary

7.1 Introduction

In this Chapter, we take the OLE-based MPC protocol devised in the previous section and extend upon it, to achieve security against up to $n-1$ malicious adversaries. Our modifications result in only a small, constant increase (specifically, a multiplicative factor of 2) in the communication complexity of the original scheme; thereby maintaining an efficient communication complexity of $\mathcal{O}(n \log q)$, where q is the characteristic of the finite field computations are performed in. We also take a deeper look at the current literature, particularly focussing on similar MPC models in which participants are assigned some correlated information before the actual MPC protocol.

Many practical MPC protocols have been presented within the literature (e.g., [48]). Most (if not all) of these protocols have overcome the prohibitive communication requirements of the early results by utilising a preprocessing phase that takes place before the actual MPC itself. This paradigm was first discussed by Beaver [4], and was known as commodity-based cryptography. More recently, Ishai et al. [68] generalised Beaver's model as the "correlated randomness" model. The basic premise involves splitting the MPC up into two phases, summarised below:

1. **Offline/Preprocessing Phase:** In this phase participants collectively compute or are assigned some correlated, random information (i.e., not related to the MPC or any participant's inputs) which is to be used later in the online

phase. In the commodity-based model of Beaver, this information is known as a commodity, and participants purchase or are assigned this information by a neutral third party.

2. **Online Phase:** The online phase is where the actual MPC is computed. This is done in an efficient manner using information theoretic primitives. The reason that this can be done so efficiently is due to the information computed in the preprocessing phase.

Generally, the correlated randomness computed in the preprocessing phase allows participants to efficiently compute multiplications in the online phase. This is because, in typical MPC, computing multiplications is an expensive operation. The reason being that in an MPC protocol each participant utilises a secret sharing scheme [97] to securely distribute shares of their input value among the other participants. Participants can then collectively carry out any computations on their assigned shares. Computing linear operations is both efficient and private (i.e., no need for communication). However, this is not the case for multiplication, which often requires a prohibitive amount of communication among participants. So, to avoid this bottleneck the bulk of the computation can be designated to the preprocessing phase.

7.1.1 Background

The MPC scheme described in Chapter 6 achieves a communication complexity of $\mathcal{O}(n \log q)$, where q is the characteristic of a finite field, $\mathbb{F}_q$ in which computations are performed in. The protocol focuses on the online phase, abstracting the preprocessing step away by assuming the presence of a mutually trusted third party (the initialiser) who distributes correlated randomness among participants. This assumption is prevalent throughout the field of MPC in the correlated randomness model as it is both a practical assumption that mirrors the real-world (e.g., vendors

selling 'commodities' via the internet [5]) and it simplifies research considerably, allowing for results that focus on improving efficiency and communication complexity in the online phase. Additionally, we note that the initialiser can always be replaced by a preprocessing phase in which participants compute the correlated randomness themselves, as is the case in [48, 47, 72, 79, 73].

Our original MPC protocol is extremely efficient for a particular type of function that contains minimal multiplications on 'nested additions'. Specifically, functions with minimal components of the form $x_a(x_b + x_c)$. This result is due to a restriction which requires participants to firstly compute all multiplications, before any linear operations can be performed. As such, in the example specified, participants would have to compute two multiplications ($x_a x_b$ and $x_a x_c$) rather then just the one, as would be the case in a variety of other MPC protocols. Furthermore, this protocol can only be performed by participants who directly hold their private information, i.e., the actual computation must be performed by the actual parties themselves, rather than designating the MPC among a set of third parties via secret sharing or some other such mechanism.

Nevertheless, even with these restrictions this result is still relevant and practical for functions without multiplication on nested addition. A concrete example of functions that take this form can be drawn from the field of machine learning. Protocols used in this field, such as linear regression can often be boiled down to simple linear algebra operations, such as computing the inner product and matrix multiplication, both of which require no nested multiplication and can therefore be computed efficiently by their protocol.

The backbone of this efficient MPC scheme is, of course, the oblivious linear evaluation (OLE) protocol which is based on [62]. The OLE protocol allows for security against a semi-honest adversary. To reiterate what was stated in Chapter

1, a stronger adversary, in which corrupted participants are able to deviate from the protocol (i.e., submitting arbitrary values, lying etc.), is known as a malicious adversary.

7.1.2 Contribution

In this work we build upon the previously presented OLE based MPC protocol and provide security against a malicious adversary. Specifically, we provide a mechanism that allows participants to check the validity of the output, letting an honest party detect foul play with high probability. As with the original protocol, our modified MPC scheme operates in a simple finite field of $\mathbb{F}_q$, allowing for the evaluation of arithmetic functions, similarly, our adapted protocol also obtains information theoretic security; meaning that we make no assumptions about the computational power of the adversary (i.e., an 'unbound' adversary with unlimited computational power).

Our result achieves the same communication efficiency of the original protocol, allowing participants to privately evaluate a multiplication with communication complexity $\mathcal{O}(n \log q)$. Furthermore, by utilising correlated randomness (as per the original semi-honest scheme) we are able to achieve security against a majority of $n-1$ participants. In the next section we contrast our result against similar protocols within the correlated randomness field.

7.1.3 Comparison to Previous Results

An abundance of research has been carried out on maliciously secure MPC with a preprocessing phase (e.g., [9, 48, 47, 72, 79, 73]), however most of these protocols make use of computational assumptions in the preprocessing phase and/or make use of black-box style functionalities or commitment schemes in the online phase that, in turn, rely on or cannot be realised efficiently without computational assumptions.

As such, whilst these protocols make use of efficient information theoretic primitives, they do not, in the strictest sense, achieve information theoretic security. The bulk of the work in the correlated randomness model that does achieve information theoretic security against a malicious adversary can be divided into four categories, commodity-based MPC, boolean circuit MPC, privacy preserving protocols, and Beaver-based protocols.

Commodity-based MPC Similar, yet disparate to the framework we utilise, is the notion of dividing the job of the initialiser among a set of non-cooperating "commodity" servers who may or may not be corrupt. This model protects against the case in which the initialiser is compromised and/or produces untrustworthy data.

However, a necessary side-effect of this is a much higher communication complexity in the pre-processing phase, when each participant must contact multiple servers to generate their required information. The work of [46] and [98] considers the two party and n-party cases respectively.

Boolean Circuit MPC There are two general methods by which a function is represented and computed via MPC, as a boolean circuit, or as an arithmetic circuit over a finite field. Whilst boolean circuit based MPC is more efficient for certain types of functions, arithmetic/finite field based MPC tends to generalise for arbitrary functions better [26]. In the general field of MPC we note that the majority of work tends to lean towards arithmetic based MPC. However, to the best of the author's knowledge, in the correlated randomness model with a malicious adversary (in particular, purely information theoretic schemes that utilise an initialiser) there seems to have been a greater focus on boolean circuit based MPC protocols, with minimal, if any work on arithmetic circuits.

Blier et al. [12] proposed an interesting protocol that considers the case where

the initialiser can supply incorrect information, constructing a boolean circuit based MPC protocol that can tolerate an initialiser sending misleading information to participants. Damgård et al. [49] presented a construction that uses a trusted initialiser to efficiently compute an arbitrary boolean circuit, even with a dishonest/corrupted majority of participants (up to $n-1$). Finally, the ground-breaking work conducted by Couteau [37] demonstrates that both arithmetic and boolean circuits can be evaluated with communication complexity sublinear to the size of the actual circuit/function itself. Couteau's protocol works on specific types of "layered" circuits and achieves malicious security for boolean circuits, but only semi-honest security for arithmetic based MPC.

Although these results are both efficient and secure, arithmetic based MPC in this model is a desirable outcome due to the fact that "unlike Boolean circuits, arithmetic circuits allow natural computations on integers to be expressed easily and efficiently" [72].

Privacy Preserving Protocols We differentiate privacy preserving protocols (PPP) from MPC protocols by delineating PPPs as protocols in which only a specific function is computed. There are many such protocols in this field and we detail a few here that utilise an initialiser (i.e., commodity-based, correlated randomness, etc.).

Works such as [54] utilise a trusted initialiser to compute the inner product of two private vectors. Whilst in [32] a matrix multiplication scheme based on correlated randomness serves as the backbone for an efficient privacy observing linear regression protocol. Of a similar note is the work conducted in [51] which produces two party, machine learning classification protocols. Many more such works exist in the literature, particularly in the semi-honest setting and whilst all such works tend to be extremely efficient, they are only efficient for specific tasks and thus, cannot

be generalised to any given function like an MPC protocol.

Beaver's Triples As mentioned previously, to the best of the author's knowledge, the vast majority of arithmetic based MPC protocols utilise a method known as 'Beaver's Triples' [4] to compute multiplication. In fact, all of the MPC schemes mentioned at the beginning of this section (the MPC protocols with preprocessing) also utilise Beaver's triples. A detailed description of this method is given in the previous Chapter. We reiterate that, due to the need to reconstruct some intermediary values, the communication complexity of this protocol is $\mathcal{O}(n^2 \log q)$. In fact, even the efficient matrix multiplication method given in [32] has a communication complexity bound by n^2, a result of their scheme being based on Beaver's triples [50].

Our scheme does not require the reconstruction of any values and as such a participant is able to compute a share of a multiplication in constant time. In fact, as per the original semi-honest protocol, to compute a given share a participant need merely exchange messages with just one other designated participant. Resulting in a total communication complexity of only $\mathcal{O}(n \log q)$

Our Work In light of the work mentioned above, our protocol achieves the following results:

- **Not limited to boolean circuits:** Generalises to a simple arithmetic function/circuit based on the finite field $\mathbb{F}_q$.
- **Security against up to $n-1$ participants:** Our scheme achieves security against a dishonest majority of $n-1$ participants.
- **Non-Beaver based:** By not utilising Beaver's triples to compute multiplications we manage to avoid the extra communication overhead associated

with reconstructing intermediary values.

In the next section we lay the foundation of our protocol by describing some preliminary information.

7.2 Preliminaries

The original *OLEMult* protocol given in the previous Chapter utilised Shamir's secret sharing scheme. For the sake of efficiency we have chosen to substitute this with the simple additive secret sharing scheme. To share a value, x_j, using this scheme, participant P_j would compute $n-1$ random shares: $x_{j_1}, \cdots, x_{j_{n-1}}$. The last share could then be computed as:

$$x_{j_n} = x_j - \sum_{l=1}^{n-1} x_{j_l}$$

A given participant, P_k is then assigned the share x_{j_k}. All computations are performed in the finite field $\mathbb{F}_q$ where q is a prime number.

As with the previous Chapter, the scheme assumes the presence of a neutral third party, known as the initialiser, who distributes information to the participants in the offline phase. Furthermore, we take the standard MPC assumption that there exists private channels between all participants. The full, modified, *OLEMult* protocol is given in Figure 7.1.

Input: P_i has x_i and P_k has the share x_{j_k}, of x_j. **Output:** P_k obtains the share γ_k, of γ where $\gamma = x_i \cdot x_j$.

Setup The initialiser privately sends:

1. A set of $n-1$ random polynomials, $S_{i_k}(x)$, of degree at most 1 to P_i where $k = 1, \cdots, n$ and $k \neq i$.
2. A random value, d_k and the value $g_k = S_{i_k}(d_k)$ to every participant P_k.

Computation P_i privately computes:

- n shares of the value 0, we will denote this value as the mask $0_i = \sum_{l=1}^{n} 0_{i_l}$.
- The multiplication polynomial, $f_i(x) = x_i \cdot x$.

Each P_k then privately executes the following steps with P_i:

1. P_k sends the value $l_k = x_{j_k} - d_k$ to P_i.
2. P_i then computes and sends to P_k the polynomial $V_k(x)$ which is computed as:

$$V_k(x) = 0_{i_k} + f_i(x + l_k) + S_{i_k}(x)$$

3. P_k computes his share of γ as $\gamma_k = V_k(d_k) - g_k$.

Figure 7.1 : *OLEMult* Protocol [31] with Additive Secret Sharing Scheme

7.3 General Model

In this section we define the model of communication as well as the security properties of our MPC protocol. We assume the usual model of communication present within many MPC protocols, that is, synchronous, secure, and authenticated private channels exist between each of the participants. However, we do not assume (or need) a broadcast channel. Due to the fact that the multiplication scheme can

only be executed on directly held input values, all multiplications must be computed first (before any other type of computation). As a result of this, the general MPC protocol can be divided into five phases:

1. **Setup Phase:** The initialiser privately assigns some multiplication data to each of the participants, as per *OLEMult*. In addition to this he also assigns participants shares to some verification data as well as some random sharing data.

2. **Sharing Phase:** Participants use the sharing data assigned to them to distribute shares of their input values among the other participants.

3. **Multiplication Phase:** In this phase all of the necessary multiplications are carried out utilising the *OLEMult* protocol. In order to maintain security each participant executes *OLEMult* twice, once with his share and the other time with the verification data assigned by the initialiser.

4. **Addition Phase:** All linear operations are privately computed in this phase, with no need for communication between participants.

5. **Verification Phase:** Once all of the computation has taken place participants simultaneously send their shares of both the output and the verification data to each of the others. Participants can then verify the validity of the output by reconstructing both the output and the verification data and then running a simple computation.

Our model of security is identical to the security with abort notion, in that the adversary is able to abort the protocol at any time. This can be achieved by having the participants under his control simply not participate in the protocol. Another way of achieving this is to have a corrupted participant raise a false complaint about

an honest participant, i.e., lie about an honest participant's behaviour. However, if the adversary does instead simply try to introduce an error into the output then, with high probability the honest participants will be able to detect this. This leads to the following definition in which we describe our requirements for a secure MPC scheme:

Definition 9. *A (t, n, ϵ)-MPC scheme is a (t, n) threshold MPC scheme (Definition 8) in which, upon completion of the protocol, a set of $n - t$ honest participants are able to detect the aberrations or malicious behaviour of an adversary (controlling the remaining t participants) with probability $1 - \epsilon$. Specifically, they are able to detect if the adversary has not followed the protocol correctly and attempted to force reconstruction of an incorrect output.*

This model of security may seem somewhat lax and low, however it is a very practical notion that is identical to the security with abort paradigm utilised in many efficient and well known MPC protocols present within the literature [9, 48, 72, 53]. The main idea behind this model being that, in the event of a failure (i.e., the MPC protocol halts) it is relatively easy to simply start the protocol again (due to the high level of efficiency). Furthermore, in a real-world scenario most participants will wish to compute the actual output of the MPC and hence, will be unlikely to cheat, especially in the face of detection.

7.4 Secure OLE-Based MPC

In this section we describe our MPC protocol and show that any errors introduced by the adversary can be detected with high probability. The method by which this detection takes place is via the utilisation of two information theoretic message authentication codes (MAC). For a given value S and two keys, α and β, the particular MAC functions utilised in our protocol are given below:

1. $M_\alpha(S) = \alpha \cdot S + \beta$

2. $M_\beta(S) = \beta \cdot S$

During the setup phase the initialiser will randomly choose both α and β. Shares of these keys will be assigned to each of the participants, however the actual values themselves are only revealed at the end of the protocol, during the verification phase. When a given participant shares his input value, he distributes not only shares of the actual value, but also shares of his input under the specific MAC functions above. For instance, when P_j shares his value, x_j (for $j = 1, \cdots, n$) we can represent it as:

$$[[x_j]] = \Big((x_{j_1}, \cdots, x_{j_n}), (M_\alpha(x_j)_1, \cdots, M_\alpha(x_j)_n), (M_\beta(x_j)_1, \cdots, M_\beta(x_j)_n) \Big)$$

Where a given participant P_k gets the share $[x_j]_k = \big(x_{j_k}, M_\alpha(x_j)_k, M_\beta(x_j)_k\big)$, for $k = 1, \cdots, n$ and each value is shared utilising the additive scheme, such that:

- $x_j = \sum_{l=1}^{n} x_{j_l}$
- $M_\alpha(x_j) = \sum_{l=1}^{n} M_\alpha(x_j)_l$
- $M_\beta(x_j) = \sum_{l=1}^{n} M_\beta(x_j)_l$

Below we describe the setup phase of our protocol, in which the initialiser privately distributes information among the participants. We note that all computations are carried out in the finite field $\mathbb{F}_q$ where q is a prime number.

7.4.1 Setup

In the setup phase of our protocol the initialiser carries out the setup phase depicted in Figure 7.1. In addition to this, he also chooses two random keys, α and β. Lastly the initialiser assigns some randomly shared values to each participants. We denote these values as random input values. Each participant is given a set of these values (one for each of his potential private inputs) whilst every other

participant is given a share of each value. Shares of each value under each of the MAC functions and the aforementioned keys are also assigned to each participant. The full setup phase is given in Figure 7.2.

Setup The initialiser distributes to all participants:

1. Sufficient multiplication information in the form of random polynomials and points on those polynomials (the setup phase in Figure 7.1).

2. Shares of two randomly chosen keys, α and β, such that each participant, P_j for $j = 1, \cdots, n$, receives the shares α_j and β_j respectively.

3. A set of random values, one for each input value held by a given participant. Specifically he assigns a value, r to P_j whilst each other participant, P_k for $k = 1, \cdots, n$, (including P_j) is given a share of this value, r_k. Shares of the MACs of this value are also given out, such that each P_k is given:

   (a) The share $M_\alpha(r)_k$, corresponding to the value $M_\alpha(r) = r \cdot \alpha + \beta$

   (b) The share $M_\beta(r)_k$, corresponding to the value $M_\beta(r) = r \cdot \beta$

Overall giving a set of shares of the form $[r]_k = \big(r_k, M_\alpha(r)_k, M_\beta(r)_k\big)$ along with shares of both keys, β_k and α_k.

Figure 7.2 : The setup phase of the MPC protocol, run by the initialiser

Although each P_j is given the value of r he does not know the value of either $M_\alpha(r)$ or $M_\beta(r)$, thus ensuring that no participant is able to compute any of the MAC keys. We also note that if the input (or the amount of input values) of each participant is not known ahead of time then we can simply assign each participant shares of the r values. During the sharing phase, when the inputs are known, each participant can then open a specific r value by having the other participants send them their corresponding shares. This is evident in the next section which describes

the sharing phase of the protocol.

7.4.2 Sharing

In this phase each participant uses the random input values, assigned to him by the initialiser, to distribute shares of his private input values. The specific protocol used to accomplish this is similar to the technique used in the well known 'SPDZ' protocol [48]. However, where they simply distribute shares under one MAC we require each participant to distribute shares under both of our previously described MAC functions. This is shown in Figure 7.3.

Sharing

1. To share his input P_j takes one of the assigned random values, r, and sends to each participant the value $\tau = x_j - r$.

2. Each participant, P_k (with the share $[r]_k = \big(r_k, M_\alpha(r)_k, M_\beta(r)_k\big)$) then computes his share of x_j along with shares of the corresponding MAC values, $M_\alpha(x_j)$ and $M\beta(x_j)$, as follows:

   (a) A designated participant, say P_1 computes his share as $x_{j_1} = r_k + \tau$, each of the other participants compute their shares as $x_{j_k} = r_k$.

   (b) $M_\alpha(x_j)_k = M_\alpha(r)_k + \tau \cdot \alpha_k$.

   (c) $M_\beta(x_j)_k = M_\beta(r)_k + \tau \cdot \beta_k$.

Figure 7.3 : Secure Share Distribution Protocol

7.4.3 Multiplication

During this phase of the MPC, the participants collaborate to compute shares to any multiplications that are needed. This is done utilising two iterations of the *OLEMult* protocol given in section 7.2. Specifically, a given participant, P_k is able

to compute a share, $[\gamma]_k = \big(\gamma_k, M_\alpha(\gamma)_k\big)$, where $\gamma = x_i \cdot x_j$ (as per section 7.2) and $k = 1, \cdots, n$. The full explanation of this is given in Figure 7.4.

Multiplication To compute shares of the multiplication $\gamma = x_i \cdot x_j$ each participant, P_k, along with P_i, executes *OLEMult* (Figure 7.1) twice to compute:

1. γ_k, which is a share of γ. This is a straightforward execution of the multiplication protocol, with P_k using x_{j_k} and P_i using x_i as their respective inputs.

2. $M_\alpha(\gamma)_k$, which is a share of $M_\alpha(\gamma)$. To compute this share P_k executes the multiplication protocol with P_i, however, this time P_k uses $M_\alpha(x_j)_k$ as his input, whereas P_i uses x_i, as before. What P_k receives after executing the protocol with P_i is a share, denoted as θ_{i_k}, which corresponds to the value $\theta_i = \gamma \cdot \alpha + \beta \cdot x_i + 0_i$. Where 0_{i_k} is a share of the mask 0_i, as per Figure 7.1. P_k then privately computes his share as follows:

$$M_\alpha(\gamma)_k = \theta_{i_k} - M_\beta(x_i)_k + \beta_k$$

The final output being that P_k has the share $[\gamma]_k = \big(\gamma_k, M_\alpha(\gamma)_k\big)$. If a given participant does not respond with an input then a complaint is raised and the protocol halts. For example, if P_i does not submit the output of *OLEMult* to P_k then P_k raises a complaint (sending a message to each of the other participants) which halts the protocol.

Figure 7.4 : *SecOLEMult*: Secure Multiplication Protocol

In the first iteration of *OLEMult*, P_k executes the protocol with P_i to compute γ_i. The second iteration allows P_k to compute $M_\alpha(\gamma)_k$. This is done by having P_k supply the share $M_\alpha(x_j)_k$ as his input into *OLEMult*, and then utilising the output in a private computation to compute $M_\alpha(\gamma)_k$.

7.4.4 Addition

Following the multiplication phase, participants are now ready to privately compute all linear functions, such as addition and multiplication by a constant. Each participant computes a share of the output (i.e., the addition or multiplication by a constant) as well as a share of the output under the MAC function $M_\alpha()$. This is shown in Figure 7.5.

Linear Operations Let S and W be two shared values, such that each participant P_k (for $k = 1, \cdots, n$) has the shares $[S]_k = \big(S_k, M_\alpha(S)_k\big)$ and $[W]_k = \big(W_k, M_\alpha(W)_k\big)$. Then:

1. To compute shares of the addition $\gamma = S + W$, each P_k privately computes $\gamma_k = S_k + W_k$ and $M_\alpha(\gamma)_k = M_\alpha(S)_k + M_\alpha(W)_k - \beta_k$.

2. To compute shares to the addition $\gamma = S+Z$, where Z is a constant, a designated individual participant, P_j (can be chosen at random or by any given method) privately computes: $Z + S_j$. To compute a share to the MAC value each P_k privately computes:
$$Z \cdot \alpha_k + M_\alpha(S)_k$$

3. Lastly, to compute shares to $\gamma = S \cdot Z$ (as before, Z is a constant) each P_k privately computes: $Z \cdot S_k$ and $Z \cdot M_\alpha(S)_k - \beta_k(Z - 1)$.

Figure 7.5 : Computing linear operations

We note that, rather than trust just one participant to compute $Z + x_{i_j}$ (P_j in Figure 7.5) the participants can instead publicly divide Z into n shares and add these shares to each of their shares of S. This may be preferable to just trusting one participant, especially if Z is known ahead of time.

7.4.5 Output and Verification

In this final phase of the protocol the output is reconstructed and checked for errors. This involves reconstructing both MAC keys, the output, and the MAC of the output. Following this, a simple computation is run that checks to see if the output is consistent with its MAC value. The full protocol is given in Figure 7.6.

Reconstruction and Verification

1. To check the validity of a given output value, γ, participants reconstruct the values: γ, $M_\alpha(\gamma)$, β and α. Reconstruction is done by simply having each of the participants simultaneously send their shares of the given values to each of the other participants. If a given participant does not send a share then the protocol aborts.

2. They take γ to be valid if $\gamma \cdot \alpha + \beta - M_\alpha(\gamma) = 0$. If this is the case then γ is taken as the output. Otherwise, cheating has occurred and the protocol aborts, outputting $\perp$.

Figure 7.6 : Secure reconstruction and verification protocol

7.4.6 Security of the Protocol

Here, we informally show that the OLE-based MPC scheme is secure against a malicious adversary that can control $n-1$ of the participants. Specifically, we show that even if such an adversary introduces an error into the computation then the participants will be able to detect this with probability $\epsilon = 1 - \frac{1}{q}$ where q is the size of the finite field used for computation. We first begin by stating that the protocol has information theoretic privacy. This is due to the privacy of the *OLEMult* protocol and the additive secret sharing scheme.

Theorem 15. *The proposed MPC protocol is perfectly private, in the information*

theoretic sense.

Proof. The proof of this is analogous to proving the security of the semi-honest *OLEMult* protocol, given in [31]. □

It remains to show that a set of $n-1$ participants cannot, with high probability, fool an honest participant into reconstructing an incorrect output.

Theorem 16. *The proposed MPC protocol is a secure* $(n-1, n, 1/q)$*-MPC protocol, as per Definition 9.*

Proof. To show this we must first examine the potential errors that the adversary can introduce into a given output. As per the work presented by Genkin et al. in [58], we classify two types of errors:

1. Additive: An additive error is simply a constant that is added to the output of the MPC protocol. For example, consider an output S. If the adversary successfully introduced an additive error, Δ_+ he would force reconstruction of $S' = S + \Delta_+$, where $\Delta_+ \neq 0$.

2. Multiplicative: The same principle applies to a multiplicative error, except here $S' = S \cdot \Delta_\times$, where $\Delta_\times \neq 1$.

Although it is possible to frame both of these types of errors as equivalent we prefer to tackle them separately, in much the same way that an MPC protocol has separate protocols for addition and multiplication.

Due to the nature of the additive secret sharing scheme a multiplicative error can be introduced at any given time. However a multiplicative error can only be successfully introduced (i.e., the error is not detected at the end of the MPC) during

the multiplication phase. We shall first show that any given undetectable (multiplicative or additive) error can only be introduced during the multiplication phase with the low probability of $1/q$.

Suppose that, during the *SecOLEMult* protocol (Figure 7.4) a set of $n-1$ participants, including P_i and P_j, are corrupted by the adversary. During the first iteration of the protocol they are able to substitute the actual output of *OLEMult* (originally $\gamma = x_i \cdot x_j$) with $\gamma' = \gamma \cdot \Delta_\times$. They will do the same with the output of the second iteration of *OLEMult*, resulting in the value $\theta'_i = \theta \cdot \delta_\times$, where $\theta_i = \gamma \cdot \alpha + \beta \cdot x_i + 0_{i_k}$. In order to successfully introduce an undetectable error the adversary would need to solve the equation:

$$\gamma' \cdot \alpha + \beta + \Delta_+ = \theta'_i - M_\beta(x_i) + \delta_+ + \beta$$

Where Δ_+ and δ_+ are additive errors introduced by the adversary. This equation can be rewritten as:

$$\gamma\Delta_\times\alpha + \beta + \Delta_+ = \gamma\alpha\delta_\times + \beta\delta_\times x_i - x_i\beta + \beta + \delta_+$$

Which simplifies to:

$$\gamma\Delta_\times\alpha + \Delta_+ = \gamma\alpha\delta_\times + x_i\beta(\delta_\times - 1) + \delta_+$$

It is easy to see that the above equation can only be solved if β is known. Since this value is random and unknown to the adversary (due to the additive secret sharing scheme) the adversary cannot solve the equation and hence, cannot introduce an undetectable error into the output during the multiplication phase, except with probability $1/q$ (i.e., the adversary guesses the correct value of β). It remains to prove that the adversary has the same chances of cheating the verification procedure (Figure 7.6).

As mentioned, the adversary is able to introduce an additive error into the output at any given time. In order to successfully cheat he will wish to force reconstruction

of a given output, $S' \neq S$, where S is the original output that would be reconstructed if no errors were introduced. Let $S' = S + \Delta_+$, to succeed in cheating the adversary must fool the verification procedure, to do this he will also introduce an additive error to the MAC value of the output, $M_\alpha(S)$. As such, denote the adversary's modified MAC as $M_\alpha(S)' = M_\alpha(S) + \delta_+$. To successfully introduce an error that goes undetected upon the completion of the protocol he will need to solve the following equation:

$$S' \cdot \alpha + \beta + \kappa = M\alpha(S)'$$

Where κ is the additive error introduced on any of the other reconstructed values. This equation then becomes:

$$\alpha(S + \Delta_+) + \kappa = \alpha \cdot S + \delta_+$$

Which simplifies to:

$$\alpha\Delta_+ + \kappa = \delta_+$$

Similar to before, to solve the above equation the adversary needs to know the value of α which, just like β, is perfectly hidden and random (due to the additive secret sharing scheme). As a result of this, the probability of successfully introducing an error into the output is $1/q$ (again, the adversary must simply guess). Therefore the proposed protocol is a secure $(n-1, n, 1/q)$-MPC scheme, as per Definition 9. □

The formal simulation-based proof of this is given in the next section.

7.5 Security of The OLE-Based MPC Protocol

In this section we prove the security of our MPC protocol utilising the simulation-based paradigm in the stand-alone security model [77]. We operate in the stand-alone model (rather than say, the universally composable model) due to the niche and specialised nature of our protocol. To clarify, our protocol can only evaluate

certain types of arithmetic circuits (as shown in section 7.6), as such the main use and purpose of our protocol will be to compute specific and specialised problems in a stand-alone, "one off fashion". For this reason, we feel that the stand-alone model of security suffices for our purposes.

An outline of our overall protocol is given in Figure 7.7 below. Note that we do not include the input phase as the initialiser is assumed to be a neutral honest third party, and thus cannot be corrupted. We also define an ideal functionality, $\mathcal{F}_{online}$ (Figure 7.8) in which an ideal world third party, $\mathcal{F}$ carries out the ideal world functionality of our protocol, alongside a simulator, $\mathcal{S}$ who simulates the dishonest participants (i.e., the adversary). In order to prove the security of our protocol we require that the output of the ideal and real world protocols, for any given set of fixed inputs, are indistinguishable, except with low probability.

An excellent, more formal definition of stand-alone, simulation-based security is given by Backes et al. in [2]. For completeness we present their definition below:

"*In the stand-alone model, a protocol* Π *securely implements an ideal function* $\mathcal{F}$ *if for every set of corrupted parties* C *and for every adversary* $\mathcal{A}$ *there is a simulator* $\mathcal{S}$ *such that the families of random variables* $REAL_{\Pi,\mathcal{A},x}$ *and* $IDEAL_{\mathcal{F},\mathcal{S},x}$ *are indistinguishable in the security parameter* k *for all inputs* $x = (x_1, ..., x_n)$*. Here* $REAL_{\Pi,\mathcal{A},x}$ *is the output of the adversary* ($\mathcal{A}$) *and of the uncorrupted parties in the following interaction:*

The uncorrupted parties $P_i \notin C$ *get input* x_i*. Then the parties interact as prescribed by the protocol* Π*. The adversary controls the corrupted parties, i.e., he can send messages in the name of a party* $P_i \in C$ *and receives all messages for parties* $P_i \in C$*. Similarly,* $IDEAL_{\mathcal{F},\mathcal{S},x}$ *consists of the output of the simulator* $\mathcal{S}$ *and of the results of the function* $\mathcal{F}$*. Here the inputs of* $\mathcal{F}$ *corresponding to the uncorrupted parties are chosen according to* x*, and the inputs of the corrupted parties are entered*

by the simulator."

For our purposes the security parameter k, is the error probability previously defined in section 7.4.6 (ϵ). As such, security for our protocol is achieved if, except with probability ϵ, $REAL_{\Pi,\mathcal{A},x} \approx IDEAL_{\mathcal{F},\mathcal{S},x}$. The proof of this is given below, whilst the overall protocol as well as the ideal functionality (in addition to the simulator's input) are given in Figures 7.7 and 7.8.

Proof. Both $\mathcal{A}$ and $\mathcal{S}$ are able to terminate the protocol at any time, $\mathcal{S}$ does this by simply sending a message to $\mathcal{F}$ whilst $\mathcal{A}$ simply has one of the participants under his control not send a message or raise a complaint.

Communication is synchronous, thus in the verification stage of the protocol $\mathcal{A}$ can pick and choose who his dishonest participants send their output shares to. This can be achieved by simply having a dishonest participant miss (i.e., not submit anything for) the given round of communication. This particular aspect is modeled in the the $\mathcal{F}_{online}$ functionality by having $\mathcal{S}$ send to $\mathcal{F}$ a list of honest participants who are allowed to receive the output.

To model the cheating of $\mathcal{A}$, the simulator $\mathcal{S}$ introduces errors into the selected dishonest participant's input by selecting false x_i' at the start of the functionality (during sharing). We argue that this is all that is needed to model the cheating of $\mathcal{A}$ in regards to share modification. This is due to the fact that $\mathcal{A}$ cannot increase his chances of successfully cheating above what random probability (i.e., selecting his false shares at random) gives him (as per the proof in section 7.4.6). Thus, the x_i' can be picked by $\mathcal{S}$ right at the start of the protocol and it will make no difference.

To model the errors introduced into the MAC values the simulator picks three random values, Δ_α, Δ_β, and Δ_M which represents the errors introduced into α, β and $M_\alpha(\gamma)$. Again, these values are chosen randomly by $\mathcal{S}$ to simulate the real world scenario in which the adversary cannot compute "valid" errors that will pass the

verification procedure, except by random chance. We can therefore conclude that $IDEAL_{\mathcal{F},\mathcal{S},x} \approx REAL_{\Pi,\mathcal{A},x}$, with a probability of error ϵ, such that $\epsilon = \frac{1}{q}$ for the finite field $\mathbb{F}_q$ (as per section 7.4.6). □

Sharing

1. To share his input P_j takes one of the assigned random values, r, and sends to each participant the value $\tau = x_j - r$.

2. Each participant, P_k (with the share $[r]_k = \big(r_k, M_\alpha(r)_k, M_\beta(r)_k\big)$) then computes his share of x_j along with shares of the corresponding MAC values, $M_\alpha(x_j)$ and $M\beta(x_j)$, as follows:

   (a) $x_{j_k} = r_k + \tau$

   (b) $M_\alpha(x_j)_k = M_\alpha(r)_k + \tau \cdot \alpha_k$

   (c) $M_\beta(x_j)_k = M_\beta(r)_k + \tau \cdot \beta_k$

Multiplication

To compute shares of the multiplication $\gamma = x_i \cdot x_j$ each participant, P_k, along with P_i, executes *OLEMult* (Figure 7.1) twice to compute:

1. γ_k, which is a share of γ. This is a straightforward execution of the multiplication protocol, with P_k using x_{j_k} and P_i using x_i as their respective inputs.

2. $M_\alpha(\gamma)_k$, which is a share of $M_\alpha(\gamma)$. To compute this share P_k executes the multiplication protocol with P_i, however, this time P_k uses $M_\alpha(x_j)_k$ as his input, whereas P_i uses x_i, as before. What P_k receives after executing the protocol with P_i is a share, denoted as θ_{i_k}, which corresponds to the value $\theta_i = \gamma \cdot \alpha + \beta \cdot x_i + 0_i$. Where 0_{i_k} is a share of the mask 0_i, as per Figure 7.1. P_k then privately computes his share as follows:

$$M_\alpha(\gamma)_k = \theta_{i_k} - M_\beta(x_i)_k + \beta_k$$

The final output being that P_k has the share $[\gamma]_k = \big(\gamma_k, M_\alpha(\gamma)_k\big)$.

Addition

Let S and W be two shared values, such that each participant P_k (for $k = 1, \cdots, n$) has the shares $[S]_k = \big(S_k, M_\alpha(S)_k\big)$ and $[W]_k = \big(W_k, M_\alpha(W)_k\big)$. Then: To compute shares of the addition $\gamma = S + W$, each P_k privately computes:

1. $\gamma_k = S_k + W_k$

2. $M_\alpha(\gamma)_k = M_\alpha(S)_k + M_\alpha(W)_k - \beta_k$

Verification

1. To check the validity of a given output value, γ, participants reconstruct the values: γ, $M_\alpha(\gamma)$, β and α.

2. They take γ to be valid if $\gamma \cdot \alpha + \beta - M_\alpha(\gamma) = 0$. If this is the case then γ is taken as the output. Otherwise, cheating has occurred and we output $\perp$. We set $REAL_{\Pi,\mathcal{A},x}$ as the output of all participants.

If **at any point int the protocol** a given participant does not respond with an input then a complaint is raised and the protocol halts and outputs $\perp$.

Figure 7.7 : Protocol Π_{online}

Initialisation

1. Each party, P_i is assigned an input x_i which is given to $\mathcal{F}$.
2. The simulator, $\mathcal{S}$ sends to $\mathcal{F}$ a list, C of cheating participants. $\mathcal{F}$ then sends the input of each party in this list to $\mathcal{S}$.
3. The initialiser sends the random MAC values, α and β to $\mathcal{F}$.

Sending Input to $\mathcal{F}$

1. For each dishonest (cheating) participant $P_i \in C$, the simulator may select any given value x_i' and send this to $\mathcal{F}$, who then uses this in place of the actual x_i value originally assigned to P_i.
2. $\mathcal{S}$ also sends to $\mathcal{F}$ the random values Δ_α, Δ_β and δ_M.

Computing The Circuit

1. $\mathcal{F}$ computes the circuit, $f(x_1, \cdots, x_n)$ utilising the original input values from the honest participants and the x_i' values supplied by $\mathcal{S}$ for the dishonest participants to compute an output value $\gamma' = f(x_1, \cdots, x_n)$.
2. $\mathcal{F}$ also computes the following values:
   - $\alpha' = \alpha + \Delta_\alpha$
   - $\beta' = \beta + \Delta_\beta$
   - $M_\alpha(\gamma)' = \gamma \cdot \alpha + \beta + \Delta_M$

Verification

1. $\mathcal{S}$ sends to $\mathcal{F}$ a list, L composed of a subset of honest participants who are permitted to have the output.
2. $\mathcal{F}$ checks whether $M_\alpha(\gamma)'$ is equal to $\alpha' \cdot \gamma' + \beta'$. If they are equal he sets γ' as the output, otherwise $\perp$ is set as the output.
3. $\mathcal{F}$ then outputs the result for each honest party listed in L as well as each of the dishonest parties. Where $IDEAL_{\mathcal{F},\mathcal{S},x}$ is set as these output values.

At any time during the protocol $\mathcal{S}$ may abort by simply sending the message *Abort* to $\mathcal{F}$ who then outputs $\perp$.

Figure 7.8 : Functionality $\mathcal{F}_{online}$ and Simulator $\mathcal{S}$

7.6 Evaluation

In this section we evaluate the efficiency of our protocol and compare it to other MPC schemes secure against a malicious adversary*. To the best of the author's knowledge, most of the current MPC schemes within the literature utilise the well known Beaver's triples protocol [5] for the purpose of computing multiplications in MPC. The general method by which secure multiplication is conducted using this method is to utilise a MAC and secret sharing scheme, similar to the one utilised in this work. Specifically, participants carry out the multiplication protocol on their shares, and once again on their shares under the MAC [73]. For each participant to compute a share of a given multiplication a total of at least $n(n-1)$ messages must be sent. Each of these messages is composed of field elements. Therefore, if all computations are conducted in the field $\mathbb{F}_q$ then this results in a communication complexity of $\mathcal{O}(n^2 \log q)$. We note that, even if just one participant wishes to compute a share (i.e., every other participant simply sends all their reconstructed values to a designated participant) they will need to reconstruct some values; as a result of this, the communication complexity for even just one participant to compute a given share is $\mathcal{O}(n \log q)$.

In our protocol $n-1$ participants send and receive just two messages to allowing for an overall communication complexity of $\mathcal{O}(n \log q)$. Furthermore, a single participant can compute their share with a constant communication complexity of $\mathcal{O}(\log q)$. This is due to the use of the efficient *OLEMult* protocol given in figure 7.1. Our protocol simply uses their protocol twice, resulting in only a small increase in communication complexity.

We note, however, that there is a significant downside to our proposed protocol.

*We specifically compare the cost of multiplication and ignore the communication complexity of the sharing phase (which is similar to many other protocols within the literature).

Specifically, multiplication can only be carried out if one of the values is explicitly known and held by a participant. As a result of this, it is not possible to compute the multiplication of two unknown (but shared) values using our protocol. Another, previously mentioned restriction of our proposed protocol is that multiplication must be completed before addition, resulting in Beaver's function being far more efficient for functions with many multiplications on nested additions.

Conversely however, it is evident that our protocol performs significantly better at tasks such as the standard way of computing matrix multiplication and the inner product of two vectors. This is due to the absence of any multiplications on 'nested additions.' This may seem somewhat limited, however we note that many models and protocols in (privacy preserving) machine learning make extensive use of matrix multiplication. As such, our protocol could, for instance, easily be adapted for use in a privacy preserving machine learning application.

Chapter 8

Conclusion

In this thesis the three inter-related fields of secret sharing, oblivious polynomial evaluation and multi-party computation were investigated. Specifically, several new protocols were proposed, existing protocols were extended and improved upon and the flaws inherent in some of the existing literature was identified and, where possible, rectified. Although we hope that the contribution of this thesis to the field of cryptography is both relevant and of interest, there are several opportunities for future research, some of which we highlight below:

Secret Sharing As mentioned in Chapter 2, an open problem within the field of SSCD is the construction of a scheme secure under the CDV model that achieves optimum share size. Additionally, construction of a SSCD scheme secure in the OKS model that is not only optimal but also supports an arbitrary field and a small secret domain is also desirable. Closer to home, development of an OSSCD scheme that tolerates dishonest and collaborating servers would also be of interest.

OPE A topic of interest identified in the field of OPE is the development of a maliciously secure OPE scheme; either a DOPE or TOPE scheme. Currently all such information theoretic OPE protocols can only tolerate semi-honest, passive adversaries.

MPC The field of MPC is vast, with many opportunities existing for future research. Looking to the results presented here however, it would be desirable

to produce an offline phase for the MPC protocol given in Chapters 6 and 7, i.e., to phase out and replace the initialiser. Additionally, future research could look at extending the MPC in order to compute a multiplication without requiring a participant to actually be holding onto one of the input values, i.e., just have each participant hold shares of the values.

Another aspect not looked at by this thesis is the actual implementation of the protocols developed here. Indeed, the construction and real-life implementation of MPC protocols is, itself, a task worthy of much study entirely separate from the theoretical construction and evaluation that was conducted in this thesis and other such research articles.

References

[1] Araki, T., Obana, S.: Flaws in Some Secret Sharing Schemes Against Cheating. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) Information Security and Privacy: 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007. Proceedings, pp. 122–132. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[2] Backes, M., Müller-Quade, J., Unruh, D.: On the necessity of rewinding in secure multiparty computation. In: Vadhan, S.P. (ed.) Theory of Cryptography. pp. 157–173. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[3] Beaver, D.: Perfect privacy for two-party protocols. In: Proceedings of DIMACS Workshop on Distributed Computing and Cryptography. vol. 2, pp. 65–77 (1991)

[4] Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. pp. 420–432. Springer Berlin Heidelberg (1992)

[5] Beaver, D.: Commodity-based cryptography (extended abstract). In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. pp. 446–455. STOC '97, ACM, New York, NY, USA (1997)

[6] Beimel, A., Chee, Y.M., Wang, H., Zhang, L.F.: Communication-efficient distributed oblivious transfer. Journal of Computer and System Sciences 78(4), 1142–1157 (2012)

[7] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. STOC '88, ACM, New York, NY, USA (1988)

[8] Benaloh, J.C.: Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract). In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 251–260. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)

[9] Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 169–188. Springer (2011)

[10] Bishop, A., Pastro, V., Rajaraman, R., Wichs, D.: Essentially optimal robust secret sharing with maximal corruptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 58–86. Springer (2016)

[11] Blakley, G.: Safeguarding cryptographic keys. In: Proceedings of the 1979 AFIPS National Computer Conference. pp. 313–317. AFIPS Press, Monval, NJ, USA (1979)

[12] Blier, H., Tapp, A.: A single initialization server for multi-party cryptography. In: Safavi-Naini, R. (ed.) Information Theoretic Security. pp. 71–85. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

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

[14] Blundo, C., D'Arco, P., Santis, A.D., Stinson, D.R.: New results on unconditionally secure distributed oblivious transfer. In: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography. pp. 291–309. SAC '02, Springer-Verlag, London, UK, UK (2003), http://dl.acm.org/citation.cfm?id=646558.694899

[15] Bo, Y., Qinglong, W., Yunfei, C.: An efficient and unconditionally-secure oblivious polynomial evaluation protocol. In: The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007). pp. 181–184 (2007)

[16] Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., et al.: Secure multiparty computation goes live. In: International Conference on Financial Cryptography and Data Security. pp. 325–343. Springer (2009)

[17] Cabello, S., Padró, C., Sáez, G.: Secret Sharing Schemes with Detection of Cheaters for a General Access Structure. Designs, Codes and Cryptography 25(2), 175–188 (2002)

[18] Cabello, S., Padró, C., Sáez, G.: Secret Sharing Schemes with Detection of Cheaters for a General Access Structure. Designs, Codes and Cryptography 25(2), 175–188 (2002)

[19] Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of CRYPTOLOGY 13(1), 143–202 (2000)

[20] Carpentieri, M., De Santis, A., Vaccaro, U.: Size of Shares and Probability of Cheating in Threshold Schemes. In: Helleseth, T. (ed.) Advances in Cryptology — EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings, pp. 118–125. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)

[21] Cevallos, A., Fehr, S., Ostrovsky, R., Rabani, Y.: Unconditionally-Secure Robust Secret Sharing with Compact Shares. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, pp. 195–208. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[22] Chang, Y.C., Lu, C.J.: Oblivious polynomial evaluation and oblivious neural learning. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 369–384. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[23] Chang, Y.C., Lu, C.J.: Oblivious polynomial evaluation and oblivious neural learning. Theoretical Computer Science 341(1-3), 39–54 (2005)

[24] Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 11–19. STOC '88, ACM, New York, NY, USA (1988)

[25] Cheong, K.Y., Koshiba, T., Nishiyama, S.: Strengthening the security of distributed oblivious transfer. In: Australasian Conference on Information Security and Privacy. pp. 377–388. Springer (2009)

[26] Choi, S.G., Hwang, K.W., Katz, J., Malkin, T., Rubenstein, D.: Secure multiparty computation of boolean circuits with applications to privacy in on-line marketplaces. In: Dunkelman, O. (ed.) Topics in Cryptology – CT-RSA 2012. pp. 416–432. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

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

[28] Chor, B., Kushilevitz, E.: A zero-one law for boolean privacy. SIAM Journal on Discrete Mathematics 4(1), 36–47 (1991)

[29] Cianciullo, L., Ghodosi, H.: Improvements to almost optimum secret sharing with cheating detection. In: Inomata, A., Yasuda, K. (eds.) Advances in Information and Computer Security. pp. 193–205. Springer International Publishing, Cham (2018)

[30] Cianciullo, L., Ghodosi, H.: Unconditionally secure distributed oblivious polynomial evaluation. In: International Conference on Information Security and Cryptology. pp. 132–142. Springer (2018)

[31] Cianciullo, L., Ghodosi, H.: Efficient information theoretic multi-party computation from oblivious linear evaluation. In: Blazy, O., Yeun, C.Y. (eds.) Information Security Theory and Practice. pp. 78–90. Springer International Publishing, Cham (2019)

[32] Cock, M.d., Dowsley, R., Nascimento, A.C., Newman, S.C.: Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security. pp. 3–14. ACM (2015)

[33] Commonwealth of Australia, Department of Health: Covidsafe app. `https://www.health.gov.au/resources/apps-and-tools/covidsafe-app`, acessed December 2020

[34] Corniaux, C.L.F., Ghodosi, H.: A verifiable distributed oblivious transfer protocol. In: Parampalli, U., Hawkes, P. (eds.) Information Security and Privacy. pp. 444–450. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[35] Corniaux, C.L.F., Ghodosi, H.: An information-theoretically secure threshold distributed oblivious transfer protocol. In: Kwon, T., Lee, M.K., Kwon,

D. (eds.) Information Security and Cryptology – ICISC 2012. pp. 184–201. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[36] Corniaux, C. L. F., Ghodosi, H.: An entropy-based demonstration of the security of Shamir's secret sharing scheme. In: 2014 International Conference on Information Science, Electronics and Electrical Engineering. vol. 1, pp. 46–48 (Apr 2014)

[37] Couteau, G.: A note on the communication complexity of multiparty computation in the correlated randomness model. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 473–503. Springer International Publishing, Cham (2019)

[38] Cramer, R., Damgård, I.B., Nielsen, J.B.: Secure multiparty computation. Cambridge University Press (2015)

[39] Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 471–488. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[40] Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings, pp. 471–488. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[41] Cramer, R., Fehr, S., Padró, C.: Algebraic manipulation detection codes. Science China Mathematics 56(7), 1349–1358 (Jul 2013)

[42] Cramer, R., Padró, C., Xing, C.: Optimal algebraic manipulation detection codes in the constant-error model. In: Dodis, Y., Nielsen, J.B. (eds.) Theory of Cryptography. pp. 481–501. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)

[43] Crépeau, C., Morozov, K., Wolf, S.: Efficient unconditional oblivious transfer from almost any noisy channel. In: International Conference on Security in Communication Networks. pp. 47–59. Springer (2004)

[44] Damgård, I., Haagh, H., Nielsen, M., Orlandi, C.: Commodity-based 2pc for arithmetic circuits. In: Albrecht, M. (ed.) Cryptography and Coding. pp. 154–177. Springer International Publishing, Cham (2019)

[45] Damgård, I., Haagh, H., Nielsen, M., Orlandi, C.: Commodity-based 2pc for arithmetic circuits pp. 154–177 (2019)

[46] Damgård, I., Haagh, H., Nielsen, M., Orlandi, C.: Commodity-based 2pc for arithmetic circuits. In: Albrecht, M. (ed.) Cryptography and Coding. pp. 154–177. Springer International Publishing, Cham (2019)

[47] Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits. In: European Symposium on Research in Computer Security. pp. 1–18. Springer (2013)

[48] Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Advances in Cryptology–CRYPTO 2012, pp. 643–662. Springer (2012)

[49] Damgård, I., Zakarias, S.: Constant-overhead secure computation of boolean circuits using preprocessing. In: Sahai, A. (ed.) Theory of Cryptography. pp. 621–641. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[50] Dankar, F.K., Madathil, N., Dankar, S.K., Boughorbel, S.: Privacy-preserving analysis of distributed biomedical data: Designing efficient and secure multiparty computations using distributed statistical learning theory. JMIR medical informatics 7(2), e12702 (April 2019)

[51] David, B., Dowsley, R., Katti, R., Nascimento, A.C.A.: Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols. In: Au, M.H., Miyaji, A. (eds.) Provable Security. pp. 354–367. Springer International Publishing, Cham (2015)

[52] Desmedt, Y., Jajodia, S.: Redistributing secret shares to new access structures and its applications. Tech. rep., Technical Report ISSE TR-97-01, George Mason University (1997)

[53] Döttling, N., Ghosh, S., Nielsen, J.B., Nilges, T., Trifiletti, R.: Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 2263–2276. CCS '17, ACM, New York, NY, USA (2017)

[54] Dowsley, R., van de Graaf, J., Marques, D., Nascimento, A.C.A.: A two-party protocol with trusted initializer for computing the inner product. In: Chung, Y., Yung, M. (eds.) Information Security Applications. pp. 337–350. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[55] Evans, D., Kolesnikov, V., Rosulek, M.: A pragmatic introduction to secure multi-party computation. Foundations and Trends® in Privacy and Security 2(2-3) (2017)

[56] Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryp-

tology. pp. 205–210. Springer US, Boston, MA (1983)

[57] Fitzi, M., Garay, J., Gollakota, S., Rangan, C.P., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Theory of Cryptography Conference. pp. 329–342. Springer (2006)

[58] Genkin, D., Ishai, Y., Prabhakaran, M.M., Sahai, A., Tromer, E.: Circuits resilient to additive attacks with applications to secure computation. In: Proceedings of the forty-sixth annual ACM symposium on Theory of computing. pp. 495–504 (2014)

[59] Ghosh, S., Nielsen, J.B., Nilges, T.: Maliciously secure oblivious linear function evaluation with constant overhead. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 629–659. Springer International Publishing, Cham (2017)

[60] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. pp. 218–229. STOC '87, ACM, New York, NY, USA (1987), http://doi.acm.org/10.1145/28395.28420

[61] Halevi, S., Ishai, Y., Kushilevitz, E., Rabin, T.: Best possible information-theoretic mpc. In: Theory of Cryptography Conference. pp. 255–281. Springer (2018)

[62] Hanaoka, G., Imai, H., Mueller-Quade, J., Nascimento, A.C.A., Otsuka, A., Winter, A.: Information Theoretically Secure Oblivious Polynomial Evaluation: Model, Bounds, and Constructions. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) Information Security and Privacy. pp. 62–73. Springer Berlin Heidelberg (2004)

[63] Hazay, C.: Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. In: Dodis, Y., Nielsen, J.B. (eds.) Theory of Cryptography. pp. 90–120. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)

[64] Hazay, C.: Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. Journal of Cryptology 31(2), 537–586 (2018)

[65] Hazay, C., Lindell, Y.: Efficient oblivious polynomial evaluation with simulation-based security. IACR Cryptology ePrint Archive 2009, 459 (2009)

[66] Hemenway, B., Lu, S., Ostrovsky, R., Welser IV, W.: High-precision secure computation of satellite collision probabilities. In: Zikas, V., De Prisco, R. (eds.) Security and Cryptography for Networks. pp. 169–187. Springer International Publishing, Cham (2016)

[67] Hoshino, H., Obana, S.: Almost Optimum Secret Sharing Schemes with Cheating Detection for Random Bit Strings. In: Tanaka, K., Suga, Y. (eds.) Advances in Information and Computer Security: 10th International Workshop on Security, IWSEC 2015, Nara, Japan, August 26-28, 2015, Proceedings, pp. 213–222. Springer International Publishing, Cham (2015)

[68] Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Sahai, A. (ed.) Theory of Cryptography. pp. 600–620. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[69] Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer–efficiently. In: Annual international cryptology conference. pp. 572–591. Springer (2008)

[70] Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. Electronics and Communications in Japan (Part III: Fundamental

Electronic Science) 72(9), 56–64 (1989)

[71] Jhanwar, M.P., Safavi-Naini, R.: Almost Optimum Secret Sharing with Cheating Detection. In: Security, Privacy, and Applied Cryptography Engineering. pp. 359–372. Springer, Cham (Oct 2015)

[72] Keller, M., Orsini, E., Scholl, P.: Mascot: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 830–842. ACM (2016)

[73] Keller, M., Pastro, V., Rotaru, D.: Overdrive: making spdz great again. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 158–189. Springer (2018)

[74] Kurosawa, K., Suzuki, K.: Almost Secure (1-Round, n-Channel) Message Transmission Scheme. In: Desmedt, Y. (ed.) Information Theoretic Security: Second International Conference, ICITS 2007, Madrid, Spain, May 25-29, 2007, Revised Selected Papers, pp. 99–112. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[75] Li, H.D., Yang, X., Feng, D.G., Li, B.: Distributed oblivious function evaluation and its applications. Journal of Computer Science and Technology 19(6), 942–947 (Dec 2004)

[76] Lindell, Pinkas: Privacy preserving data mining. Journal of Cryptology 15(3), 177–206 (Jun 2002), https://doi.org/10.1007/s00145-001-0019-2

[77] Lindell, Y.: How to simulate it–a tutorial on the simulation proof technique. In: Tutorials on the Foundations of Cryptography, pp. 277–346. Springer (2017)

[78] Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. pp. 36–54. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)

[79] Lindell, Y., Pinkas, B., Smart, N.P., Yanai, A.: Efficient constant round multi-party computation combining bmr and spdz. In: Annual Cryptology Conference. pp. 319–338. Springer (2015)

[80] Naor, M., Pinkas, B.: Oblivious Polynomial Evaluation. SIAM Journal on Computing 35(5), 1254–1281 (Jan 2006)

[81] Naor, M., Pinkas, B.: Oblivious Transfer and Polynomial Evaluation. In: Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing. pp. 245–254. STOC '99, ACM, New York, NY, USA (1999)

[82] Naor, M., Pinkas, B.: Distributed oblivious transfer. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 205–219. Springer (2000)

[83] Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM Journal on Computing 35(5), 1254–1281 (2006)

[84] Nikov, V., Nikova, S., Preneel, B., Vandewalle, J.: On unconditionally secure distributed oblivious transfer. In: International Conference on Cryptology in India. pp. 395–408. Springer (2002)

[85] Obana, S.: Almost optimum t-cheater identifiable secret sharing schemes. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 284–302. Springer (2011)

[86] Obana, S., Tsuchida, K.: Cheating Detectable Secret Sharing Schemes Supporting an Arbitrary Finite Field. In: Advances in Information and Computer Security. pp. 88–97. Springer, Cham (Aug 2014)

[87] Ogata, W., Kurosawa, K.: Optimum Secret Sharing Scheme Secure against Cheating. In: Maurer, U. (ed.) Advances in Cryptology — EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings, pp. 200–211. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)

[88] Ogata, W., Kurosawa, K.: Oblivious keyword search. Journal of complexity 20(2-3), 356–371 (2004)

[89] Ogata, W., Kurosawa, K., Stinson, D.R.: Optimum secret sharing scheme secure against cheating. SIAM Journal on Discrete Mathematics 20(1), 79–95 (2006)

[90] Otsuka, A., Imai, H.: Unconditionally Secure Electronic Voting, pp. 107–123. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), https://doi.org/10.1007/978-3-642-12980-3_6

[91] Özarar, M., Özgit, A.: Secure multiparty overall mean computation via oblivious polynomial evaluation. In: Security of Information and Networks: Proceedings of the First International Conference on Security of Information and Networks (SIN 2007). p. 84. Trafford Publishing (2008)

[92] Peter, A., Tews, E., Katzenbeisser, S.: Efficiently outsourcing multiparty computation under multiple keys. IEEE transactions on information forensics and security 8(12), 2046–2058 (2013)

[93] Rabin, M.O.: How to exchange secrets with oblivious transfer (2005), http://eprint.iacr.org/2005/187, harvard University Technical Report 81 talr@watson.ibm.com 12955 received 21 Jun 2005

[94] Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: Proceedings of the Twenty-first Annual ACM Sym-

posium on Theory of Computing. STOC '89, ACM, New York, NY, USA (1989)

[95] Rivest, R.L.: Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Unpublished manuscript (1999)

[96] Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)

[97] Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)

[98] Smart, N.P., Tanguy, T.: Taas: Commodity mpc via triples-as-a-service. In: Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop. pp. 105–116 (2019)

[99] Tompa, M., Woll, H.: How to Share a Secret with Cheaters. Journal of Cryptology 1(2), 133–138 (Aug 1988)

[100] Tonicelli, R., Nascimento, A.C.A., Dowsley, R., Müller-Quade, J., Imai, H., Hanaoka, G., Otsuka, A.: Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. International Journal of Information Security 14(1), 73–84 (Feb 2015)

[101] Yao, A.C.: Protocols for secure computations. In: Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on. pp. 160–164. IEEE (1982)

[102] Zhu, H., Bao, F.: Augmented oblivious polynomial evaluation protocol and its applications. In: di Vimercati, S.d.C., Syverson, P., Gollmann, D. (eds.) Computer Security – ESORICS 2005. pp. 222–230. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)