Université de Montréal

Practical and Foundational Aspects of Secure Computation

par
Samuel Ranellucci

Département computer science
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en Février

2014,

.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

COM	Commitment
CRS	Common Reference String
DDH	Decisional Diffie-Hellman
Dist	Distribute
GOT	Generalized Oblivious Transfer
GVSS	Ghostly Verifiable Secret Sharing
IC	Information checking
LDS	Linear distributed Secret
MPC	Multiparty Computation
NIZK	Non-Interactive Zero-Knowledge
NSA	National Security Agency
OT	Oblivious Transfer
Rec	Reconstruct
SS	Secret Sharing
UC	Universal Composability
VOT	Verifiable Oblivious Transfer
VSS	Verifiable Secret Sharing
ZK	Zero-Knowledge

RÉSUMÉ

Il y a des problèmes qui semblent impossible à résoudre sans l'utilisation d'un tiers parti honnête. Comment est-ce que deux millionnaires peuvent savoir qui est le plus riche sans dire à l'autre la valeur de ses biens ? Que peut-on faire pour prévenir les collisions de satellites quand les trajectoires sont secrètes ? Comment est-ce que les chercheurs peuvent apprendre les liens entre des médicaments et des maladies sans compromettre les droits privés du patient ? Comment est-ce qu'une organisation peut empêcher le gouvernement d'abuser de l'information dont il dispose en sachant que l'organisation doit n'avoir aucun accès à cette information ? Le Calcul multiparti, une branche de la cryptographie, étudie comment créer des protocoles pour réaliser de telles tâches sans l'utilisation d'un tiers parti honnête.

Les protocoles doivent être privés, corrects, efficaces et robustes. Un protocole est privé si un adversaire n'apprend rien de plus que ce que lui donnerait un tiers parti honnête. Un protocole est correct si un joueur honnête reçoit ce que lui donnerait un tiers parti honnête. Un protocole devrait bien sûr être efficace. Être robuste correspond au fait qu'un protocole marche même si un petit ensemble des joueurs triche. On démontre que sous l'hypothèse d'un canal de diffusion simultané on peut échanger la robustesse pour la validité et le fait d'être privé contre certains ensembles d'adversaires.

Le calcul multiparti a quatre outils de base : le transfert inconscient, la mise en gage, le partage de secret et le brouillage de circuit. Les protocoles du calcul multiparti peuvent être construits avec uniquements ces outils. On peut aussi construire les protocoles à partir d'hypothèses calculatoires. Les protocoles construits à partir de ces outils sont souples et peuvent résister aux changements technologiques et à des améliorations algorithmiques. Nous nous demandons si l'efficacité nécessite des hypothèses de calcul. Nous démontrons que ce n'est pas le cas en construisant des protocoles efficaces à partir de ces outils de base.

Cette thèse est constitué de quatre articles rédigés en collaboration avec d'autres chercheurs. Ceci constitue la partie mature de ma recherche et sont mes contributions principales au cours de cette période de temps. Dans le premier ouvrage présenté dans cette thèse, nous étudions la capacité de mise en gage des canaux bruités. Nous démontrons tout d'abord une limite inférieure stricte qui implique que contrairement au transfert inconscient, il n'existe aucun protocole de taux constant pour les mises en gage de bit. Nous démontrons ensuite que, en limitant la façon dont les engagements peuvent être ouverts, nous pouvons faire mieux et

même un taux constant dans certains cas. Ceci est fait en exploitant la notion de ≪cover-free families ≫. Dans le second article, nous démontrons que pour certains problèmes, il existe un échange entre robustesse, la validité et le privé. Il s'effectue en utilisant le partage de secret vérifiable, une preuve à divulgation nulle, le concept de fantômes et une technique que nous appelons les balles et les bacs. Dans notre troisième contribution, nous démontrons qu'un grand nombre de protocoles dans la littérature basée sur des hypothèses de calcul peuvent être instanciés à partir d'une primitive appelée Transfert Inconscient Vérifiable, via le concept de Transfert Inconscient Généralisé. Le protocole utilise le partage de secret comme outils de base. Dans la dernière publication, nous counstruisons un protocole efficace avec un nombre constant de rondes pour le calcul à deux parties. L'efficacité du protocole dérive du fait qu'on remplace le coeur d'un protocole standard par une primitive qui fonctionne plus ou moins bien mais qui est très peu couteux. On protège le protocole contre les défauts en utilisant le concept de ≪privacy amplification ≫.

Mots Clés : Boite-noire, Transfert inconscient, Partage de Secret, Mise en Gage, Circuit Brouillé , Robustesse, Validité, Privé

ABSTRACT

There are seemingly impossible problems to solve without a trusted third-party. How can two millionaires learn who is the richest when neither is willing to tell the other how rich he is? How can satellite collisions be prevented when the trajectories are secret? How can researchers establish correlations between diseases and medication while respecting patient confidentiality? How can an organization insure that the government does not abuse the knowledge that it possesses even though such an organization would be unable to control that information? Secure computation, a branch of cryptography, is a field that studies how to generate protocols for realizing such tasks without the use of a trusted third party. There are certain goals that such protocols should achieve. The first concern is privacy: players should learn no more information than what a trusted third party would give them. The second main goal is correctness: players should only receive what a trusted third party would give them. The protocols should also be efficient. Another important property is robustness, the protocols should not abort even if a small set of players is cheating.

Secure computation has four basic building blocks : Oblivious Transfer, secret sharing, commitment schemes, and garbled circuits. Protocols can be built based only on these building blocks or alternatively, they can be constructed from specific computational assumptions. Protocols constructed solely from these primitives are flexible and are not as vulnerable to technological or algorithmic improvements. Many protocols are nevertheless based on computational assumptions. It is important to ask if efficiency requires computational assumptions. We show that this is not the case by building efficient protocols from these primitives. It is the conclusion of this thesis that building protocols from black-box primitives can also lead to efficient protocols.

This thesis is a collection of four articles written in collaboration with other researchers. This constitutes the mature part of my investigation and is my main contributions to the field during that period of time. In the first work presented in this thesis we study the commitment capacity of noisy channels. We first show a tight lower bound that implies that in contrast to Oblivious Transfer, there exists no constant rate protocol for bit commitments. We then demonstrate that by restricting the way the commitments can be opened, we can achieve better efficiency and in particular cases, a constant rate. This is done by exploiting the notion of cover-free families. In the second article, we show that for certain problems,

there exists a trade-off between robustness, correctness and privacy. This is done by using verifiable secret sharing, zero-knowledge, the concept of ghosts and a technique which we call "balls and bins". In our third contribution, we show that many protocols in the literature based on specific computational assumptions can be instantiated from a primitive known as Verifiable Oblivious Transfer, via the concept of Generalized Oblivious Transfer. The protocol uses secret sharing as its foundation. In the last included publication, we construct a constant-round protocol for secure two-party computation that is very efficient and only uses black-box primitives. The remarkable efficiency of the protocol is achieved by replacing the core of a standard protocol by a faulty but very efficient primitive. The fault is then dealt with by a non-trivial use of privacy amplification.

Keywords: Black-Box, Oblivious Transfer, Secret Sharing, Commitment scheme, Garbling scheme, Robustness, Correctness, Privacy

I dedicate this thesis to my grand-mother Elizabeth von Wartburg Knoll who has shown me infinite patience.

ACKNOWLEDGMENTS

I would to thank my grand-parents for having the patience to allow me to live with them. I would like thank my parents and my sisters for believing in me. I would like to thank Andrea, Justin and Valerie Brauer for treating me like family. I would like to thank Claude Crépeau for teaching me the fundementals of my field. I would like to thank Anne Broadbent, Stacey Jeffery, Ryo Nishimaki, David Bernardo, Jürg Wullschleger and Severin Winkler for their efforts as contributors to my papers. I would like to thank the members of my lab, Benno Salwey, Olivier Coutu, Patrick Baril Robichaud, Michael Cadillac, Dave Touchette for being good friends, correcting my work and for taking the time to understand what I'm trying to communicate to them. I would like to thank Chantal Marie Helène for correcting the french version of the abstract. I would like to thank Marc Feeley for suggesting the term building blocks for this thesis. I would like to thank Gilles Brassard and Louis Salvail for their words of wisdom. I would like to thank the people who helped me improve my work: Jürg Wullschleger, Jesper Buus Nielsen, Tore Kasper Frederisken, Serge Fehr and the many anonymous referees. I would also like to thank the people who invited me and hosted me: I would like to thank Claude Crépeau for inviting me to his workshops. I would like to thank Ivan Damgård and the Aarhus cryptography group for allowing me to attend their MPC workshop titled "theory and practice of multiparty computation" and for later inviting me to present my research to their group. I would like thank Anne Broadbent and the IQC for hosting me at Waterloo and to Dan Dandynov for letting me stay at his place during that period of time. I would like to thank the committee for correcting my thesis. Lastly, I would like to thank my supervisor Alain Tapp's without whose encouragement, guidance and focus, I would have been unable to complete this thesis.

CHAPTER 1

INTRODUCTION

The age of information has arrived. With the rise of data mining, the amount of stored information has grown tremendously : e-mails, web habits, medical databases, location tracking, etc. In addition, the ability to process such information, both algorithmically and in terms of raw processing power has increased dramatically. If used wisely, it could protect nations, grant us medical knowledge that saves lives, improve the effectiveness of businesses, save us from the annoyance of bureaucratic tasks. It could even simplify and expand the democratic process.

On the other hand, if the privacy of ordinary citizens is betrayed.The consequences to the layman could be dire. They could be prevented from travelling, lose their jobs, they can suffer humiliation and be sent to prison.

It is this potential and this fear that establishes the importance of secure computation. It is the field that studies how parties can learn the output of an agreed upon function based on their joint inputs without leaking additional information. Many of the following sections of this chapter will contain applications of secure computation. In the last section, we will discuss the utility of building blocks of secure computation.

Statistical database

Secure computation can allow us to gain medical knowledge which can save lives. The goal is to find information without revealing private data from within databases. There is a lot of data that people don't wish to share but which could be used to save and improve our lives. This information can be used to find cures, link drugs to harmful effects and even localize outbreaks of disease. A concrete example would be to compare the prevalence of heart disease in a population taking a certain drug to a control group. A secure protocol should do this without revealing which patients either take the drug or have a heart disease.

Big brother

Secure computation can be applied to allow the government to keep us safe without sacrificing our liberties. In 2013, documents leaked by Edward Snowden revealed the extent of government spying. In short, it was revealed that the National Security Agency (NSA) had unlimited access to e-mail accounts, cell phone data, browsing habits and other metadata of private citizens. Many governments have taken the position that even if illegal, it is necessary that they capture and process all this information to keep people safe. There have been many abuses, people ending up on no-fly lists, government employees spying on their potential mates. In contrast, civil organisations have demanded that privacy be respected. It may seem impossible to reconcile such contradictory positions, but secure computation may provide solutions. In particular, we can envision a system where data can be shared between a governmental agency and a non-governmental group. Any data is only released if both the agency and the group authorize its distribution. This can be done in such a way that requests stay secret but at the same time the group is able to verify that the requests either respects privacy or is properly authorized.

In addition, this could also improve people's lives. Applying for a visa, a passport, a residency permit or filling out taxes is an extremely frustrating waste of time. In some cases the government already has the information required, and in other cases it could easily acquire it. Privacy has to be maintained and so the government should not have control of that knowledge. An elegant solution would be to securely distribute all the information between the government and organizations dedicated to privacy. In addition, this information could be used to prove the innocence of people by providing reliable alibis. This would allow us to have all the advantages of *Big Brother* without living in the world of *1984* from [Orw49].

Sugar Beet auctions

Secure computation can also be applied to markets. In Denmark, farmers are contracted by the company Danisco to produce sugar beets, each farmer is contracted to provide a certain amount of sugar beets. European subsidies for sugar beets ceased. As a result, it was necessary to redistribute the production rights between farmers. The farmers had to decide how much production rights they would buy or sell for a given price. The farmers didn't want this information to be public because it would give Danisco a stronger bargaining position.

A functionality was implemented to determine the market clearing price, the price at which demand and supply are in equilibrium. This is the first practical implementation of secure computation, its full description can be found in [BCD+09].

Satelite collision prevention

Even in Space, there are even useful applications of secure computation. There are thousands of satellites in orbit. Each satellite cost millions of dollars to launch into orbit. If two satellites collide, the monetary loss is gigantic. It also generates floating debris which poses a risk to other satellites. The satellites "Iridium 33" and "Kosmos 2251" were lost due to such a collision. A trivial solution to prevent such accidents would be to simply reveal the trajectory of the satellites. Unfortunately, many parties due to security concerns are unwilling to publicize this information. A company, named Cybernetica has developed a solution using secure computation to prevent such occurrences.

Building Blocks

Secure computation can be achieved using four basic building blocks: Oblivious Transfer, secret sharing, commitment schemes, and garbled circuits. Protocols can be built based only on these building blocks from the seminal work of [Kil88] and [Yao82]. A black-box protocol is a protocol which is constructed exclusively from these primitives. These building blocks can be instantiated from a variety of computational assumptions and from information theoretic primitives such as noisy channels. If an assumption is no longer considered secure, it is easy to replace a building block instantiation with another one based on another hardness assumption.

In contrast, protocols can be constructed with the aid of specific computational assumptions. For example, many protocols are based on the discrete logarithm problem or on the decisional Diffie-Helman problem. This means that the protocol is secure only if the underlying problem is hard.

There are many disadvantages of protocols built on computational assumptions. They can suffer due to algorithmic or technological improvements. They can become completely worthless. For example, a quantum computer would render any protocol based on the hardness of factoring completely useless. Algorithmic improvement could force certain protocols

to use larger cryptographic keys. This would make the protocol less efficient.

In contrast, protocols based on these building blocks are more robust and can be easily adapted to deal with technological and algorithmic improvement. Naturally one may ask, why build protocols under specific computational assumptions? The reason is that it seems easier to build more practical protocols based on specific computational assumptions. The question is can we build protocols that are constructed solely from black-box primitives that are as efficient as those built using specific computational assumptions. The main focus of this thesis is to show that this is indeed the case.

CHAPTER 2

PRELIMINARIES

In this chapter, we will introduce important notions of secure computation. We will define a strong notion of security called Universal Composability and we will precisely explain the building blocks of secure computation.

2.1 Security

Many protocols that were once deemed "secure" were found to be completely broken when combined with other protocols. It is therefore important to define a notion of security where protocols remain secure even under composition. Universal composability from [Can01] is a paradigm for proving that a protocol is secure even under composition. It is based on simulating the trusted third party. The main feature of universal composability is the composition theorem. It guarantees that security of protocols is not compromised by running other secure protocols in parallel and that any ideal functionality can be replaced by a protocol that securely implements it.

Ideal functionality

An ideal functionality is a way of specifying the realized security properties of a protocol. It is modeled by a trusted third party that takes the input from each participant and outputs the values consistent with its specification to each party. The ideal functionality in the universal composability framework knows which parties are corrupt (see [CLOS02],[Can06]). This can be useful for modeling functionalities that leak information against a corrupt player or on adaptive corruption.

Real Model

In the real world, a reflection of reality, when considering a protocol between two players, there are three parties : the environment $\mathcal{Z}$, the honest party $\mathcal{P}$ and the adversary $\mathcal{A}$. This can easily be generalized for protocols with more parties. The environment $\mathcal{Z}$ provides some initial input to $\mathcal{A}$ and party $\mathcal{P}$. The honest party $\mathcal{P}$ will faithfully execute the protocol

π as specified and will only forward messages that are outputs of the functionality to the environment. The adversary will execute arbitrary code and relay any message that it wishes to the environment and vice-versa. $REAL_{\pi,\mathcal{A},\mathcal{Z}}$ denotes the random variable induced by running a particular $\mathcal{A}, \mathcal{Z}$. It is important to note that the adversary can be entirely removed in this setting by allowing $\mathcal{Z}$ to interact directly with the honest party in the protocol execution.

Ideal model

In the ideal world, a reflection of what we want the protocol to realize, when considering a protocol between two players, there are three parties namely $\mathcal{Z}, \mathcal{P}$ and a new party called the simulator $\mathcal{S}$. In this model, the environment starts in the same way by giving inputs to each party. The simulator will only be able to interact with the honest party through the ideal functionality $\mathcal{F}$. The simulator will also interact with the environment in the same way that an adversary in the real model would. $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denotes the random variable formed by running a particular $\mathcal{S}, \mathcal{Z}$ for a given functionality $\mathcal{F}$.

Hybrid model

The hybrid model is an extension of the real model. In addition to the real model, the parties have access to an unbounded number of instances of an ideal functionality $\mathcal{F}$. The parties can interact directly with these functionalities. We call it the $\mathcal{F}$-Hybrid model. However, $REAL_{\pi,\mathcal{A},\mathcal{Z}}$ will still be used to denote the random variable even under a $\mathcal{F}$-hybrid model.

Security definition

For a protocol to be considered secure, for every adversary there exists a simulator such that the view (the messages it receives) generated for the environment by the adversary can also be generated by the simulator. The security can come in many different flavors, it can be perfect, statistical or computational. A protocol is perfectly secure if the views are the same. A protocol is statistically secure if the views are statistically indistinguishable. A protocol is computational if no polynomial time adversary can distinguish them.

Definition 1 *A protocol π securely realizes an ideal functionality $\mathcal{F}$, if for every adversary*

$\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that, for every environment $\mathcal{Z}$,the following hold:

$$REAL_{\pi,\mathcal{A},\mathcal{Z}} \approx IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$$

Composition theorem

The composition theorem demonstrates that a protocol which is secure in the universal composability framework remains secure even when used as a subroutine or used in parralel with other protocols. Informally, if a protocol $\pi_{\mathcal{F}}$ securely realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model and that $\pi_{\mathcal{G}}$ securely realizes $\mathcal{G}$. If we create $\pi'_{\mathcal{F}}$ by substituting $\mathcal{G}$ for $\pi_{\mathcal{G}}$ in $\pi_{\mathcal{F}}$, the resulting protocol will still securely realize $\mathcal{F}$. The basic way to prove this theorem is by showing that if it weren't true then it would be possible to generate an environment that would be able to distinguish between $\pi_{\mathcal{G}}$ and $\mathcal{G}$ or between $\pi_{\mathcal{F}}$ and $\mathcal{F}$. This results in a contradiction and as such completes the proof sketch of the composition theorem.

Privacy and correctness

Intuitively, privacy is the notion that the only information any set of players should learn is what they can learn from their own inputs and the output of the functionality. Correctness is the notion that what each player receives is what the ideal functionality would have given them. Privacy and correctness are the main concerns of secure computation. In some cases privacy trumps correctness while in other it may be the opposite.

An interesting way of showing that a protocol is private but not correct is by defining a functionality f_{priv} which takes the input from all participants, sends the output to the adversary, takes an extra string w from the adversary and sends w to the honest players. Similarly, we can see a protocol that is correct but not private by defining a functionality $f_{correct}$ which takes the input from all participants, then leaks the sender's input and sends the output to all participants.

2.2 Black-box primitives

In this section, we introduce the different building blocks of secure computation. Secret Sharing, Commitment schemes, Oblivious Transfer and garbling schemes allow us to construct secure computation. Secret sharing and commitment schemes allow us to verifiy that

players acted honestly. Garbling schemes enable efficient constant-round secure two-party computation. Oblivious Transfer is the most important building block of secure computation.

Secret sharing

A secret sharing scheme ([Sha79]) is a method of generating shares of a secret. Certain subsets of shares allow the secret to be reconstructed. The subsets of shares which do not reveal the secret should give no information about the secret. The access structure defines which shares allow reconstruction of the secret. The simplest secret sharing is when the shares add up to the secret, the only way to have any information about the secret is to have all the shares. We refer to the collections that contains all set of shares that allows a receiver to construct the secret as the access structure. An important example of secret sharing is Shamir's secret sharing scheme. It is a secret sharing with parameters (t, n) where $t \leq n$. In this scheme, there are exactly n shares, any set of t shares allows reconstruction of the secret and less than t shares give no information about the secret. Verifiable secret sharing schemes are secret sharing schemes with the additional property that even if a small set of shares have been corrupted, faulty shares can be identified and thus the secret can still be reconstructed. In most contexts, a secret sharing scheme is defined in terms of players rather than shares. For our purpose, this is insufficient.

Commitment schemes

A physical implementation of a commitment scheme which may help the reader understand the primitive is as follows: In the commit phase, the sender puts his message in a safe and gives the safe to the receiver. In the reveal phase, the sender gives the combination of the safe to the receiver. The receiver can then open the safe and learn the message. This scheme is *binding* since the receiver has the safe and thus the sender cannot change the message in the safe. This scheme is *concealing* since the receiver cannot open the safe without learning the combination.

A commitment scheme [BCC88, GMW91] is a two-phase primitive between a sender and a receiver. In the commit phase, the sender first commits to a message. In the reveal phase, the sender reveals some information to the receiver which allows him to learn the message. The binding property is that after the commit stage there exists a unique value that the receiver will accept as the message. The concealing property requires that prior

to the reveal phase, the receiver should not be able to learn any information about the message or any function of committed messages. Commitment schemes can be instantiated from a variety of computational assumptions [GMW86, BCC88, GMR89, Nao91, GK96**?** , HILL99, HR07, FLM11, Lin11]. Commitment schemes can also be instantiated from a variety of information-theoretic primitives [Cré97, DKS99, WNI03, Wul09].

Commitment schemes were introduced by Manuel Blum in [Blu83]. They are useful for generating efficient protocols. They allow players to verify that the other participants acted honestly.

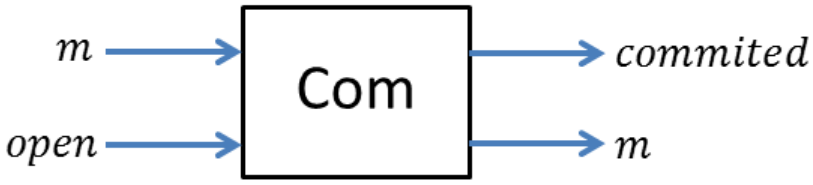

Figure 2.1 – Commitment

Oblivious Transfer

Intuitively, Oblivious Transfer (OT) is a primitive where the sender selects two messages, the receiver learns one of them, learns no information about the other message and the sender learns no information on which message the receiver chose. Oblivious Transfer was first introduced by [Rab81], originally it was a primitive that resembles an erasure channel with the additional property that the sender was oblivious to whether or not the receiver acquired the messages. The variant more commonly used for secure computation was introduced by [EGL85]. Oblivious Transfer was actually conceived earlier by Wiesner in an unpublished manuscript [Wie83] but he did not forsee its applications to cryptographic protocols. It was shown in [Cré87] that these two variants are in fact equivalent. Oblivious Transfer is a primitive that is complete for two-party computation as shown by Killian in [Kil88]. Oblivious Transfer can be instantiated from a variety of computational assumptions [BM90, FMR96, KSV07, PVW08, GH08, CKWZ13]. Oblivious Transfer can also be instantiated under a variety of information-theoretic of primitives. [CK88, CS91, Bea96, CCM98]

In contrast to Commitment schemes, Oblivious Transfer is complete for two-party computation. It is thus a more powerful primitive.

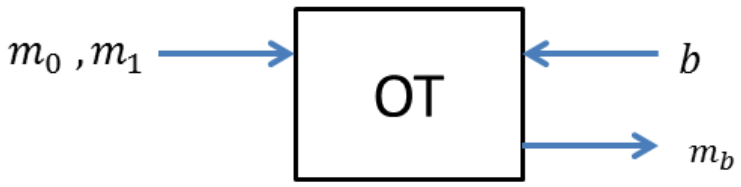

Figure 2.2 – Oblivious Transfer

Verifiable Oblivious Transfer

Verifiable Oblivious Transfer is a variant of Oblivious Transfer where the sender is committed to his inputs. This variant of Oblivious Transfer allows the receiver to verify that the sender acted honestly. Verifiable Oblivious Transfer is similar to another variant called Committing OT. Verifiable Oblivious Transfer was presented in [CC00] and [KSV07]. Committing OT appears in the works of [JS07] and [SS11].

As specified in the diagram, Alice sends two message, Bob chooses to learn one of them. This is the OT part of this primitive. Later, Alice can ask the functionality to reveal one of the messages of her choice to Bob. This is the commitment part of this functionality.

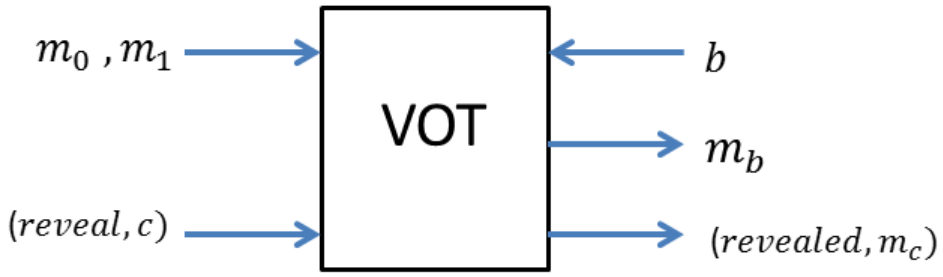

Figure 2.3 – Verifieable Oblivious Transfer

Generalized Oblivious Transfer

Generalized Oblivious Transfer is an interesting application of Verifiable Oblivious Transfer. It was first described by [IK97]. An interesting way of describing an OT is by describing the groups of messages that the receiver can get as sets in a collection. In the case of a simple OT, he can learn one of the sets in $\{\{1\},\{2\}\}$. K-out-of-N OT is an OT with a collection that contains all the sets of messages of size k or less. This mindset allows us to generalize Oblivious Transfer to its most general form. There is an important link between Generalized Oblivious Transfer and secret sharing.

Figure 2.4 – Generalized Oblivious Transfer

The works of [SSR08] and [Tas11] construct GOT by using secret sharing. The basic idea of [SSR08] is that the sender constructs a secret and generates shares for it. The sender allows the receiver for each message to either learn a share of the secret or to learn a one-time pad that will be used to decrypt a cyphertext of the message. The receiver is then supposed to forward the secret to the sender and then the sender sends the encrypted messages to the receiver.

Unfortunately both protocols are insecure. The protocols from [SSR08] and [Tas11] are insecure against a malicious sender. A malicious sender can easily break the privacy of both schemes. In the protocols, a simple OT is used and share validation is non-existent. As a result, it is possible for the sender to determine if a specific message was chosen. A concrete attack on [SSR08] will be demonstrated. An adversary wishes to learn if a receiver learns the message m_c, he selects a secret S and executes the share algorithm resulting in shares s_j. He replaces s_c by s'_c and executes the GOT protocol with those shares. As a result, if the receiver selects m_c, he will reconstruct s correctly otherwise he will reconstruct an $s' \neq s$. The attack breaks the privacy of the receiver. The same idea can be applied to attack the protocol from [Tas11].

As part of the third article, we show how to securely realize GOT with the aid of commitments and Verifiable Oblivious Transfer.

Garbling scheme

In 1982, Yao in [Yao82] generated a construction that came to be known as Yao's garbled circuit. The basic idea was to encode a circuit in such a way that by giving a party the right keys he could evaluate the circuit on a specific input and yet learn nothing about the input except what could be deduced from the circuit. This construction came to be used in many different applications each with their own divergent variants. Lindell and Pinkas were the first to prove the security of Yao's garbled circuit in the field of two party computation nearly 20 years later in [LP04]. The construction is only secure in the semi-honest model. Garbling schemes decribed in [BHR12] characterize the notions and generate definitions that unify these different variants. Garbling schemes can be used under the assumption that block ciphers act as random permutations or under the assumption of random oracles. In our case, garbling schemes allow us to instantiate constant-round secure computation that is practically efficient even against malicious adversaries. The idea behind these constructions is to generate many circuits, use one subset of circuits to verify that the circuits are well formed and use the others to evaluate the function.

Intuitively, a garbling scheme is an algorithm which takes the description of a function and outputs three functions. The first function takes any input and encrypts it, this function is called the encoding function. The second function (evaluation function) can only generate an encrypted output from an encrypted input. The third function (decoding function) takes an encrypted output and converts it to a plaintext. The evaluation function, the decoding function and the output should only reveal a limited amount of information about the garbled function.

In secure computation, garbling schemes are used to instantiate constant round secure two-party computation. A simplification of how garbling schemes are used is as follows: we will call the parties the sender and the receiver. They wish to compute f with inputs x and y respectively. The sender will first compute the function $g_x(y) := f(x, y)$ which is the currying of the function f with input x. He will then garble it and send the last two resulting functions to the receiver. The sender, which posseses the encoding function, and the receiver will use oblivious transfer so that the receiver learns his input encrypted, the sender though

is assured that the receiver does not gain additional information. The receiver thus learns the output without gaining additional information.

The diagram that follows explain that running a garbling scheme with function f, input x produces the same result as simply evaluating f on input x. Garbling scheme also guarantee that knowing F, X, d does not reveal additional information except what is specified by the leakage function and the output of the function.

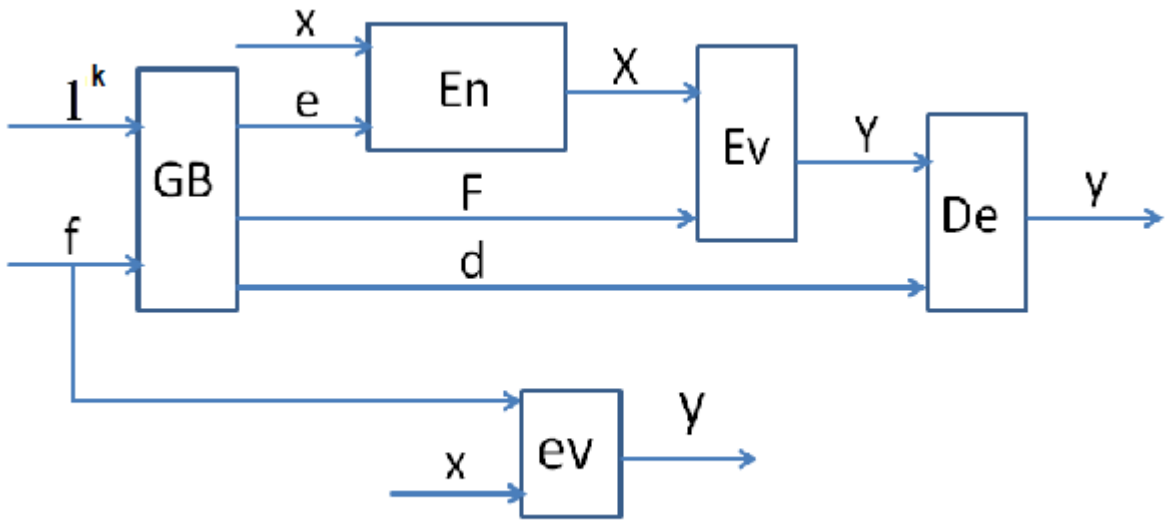

Figure 2.5 – Garbling scheme

CHAPTER 3

CONTRIBUTIONS

In this chapter, we specify the contributions of the articles listed in this thesis. The personal contributions of the author of this thesis are specified at the end of this chapter.

On the efficiency of bit commitment reductions

In the first paper, we show that any protocol that implements n instances of bit commitments with security parameter s (-log of the failure probability) requires ns instances of Oblivious Transfer or calls to a noisy channel. In particular, it is impossible to achieve a constant rate. This is in contrast to Oblivious Transfer where a constant rate reduction is possible We then show that by restricting the way in which the bit commitments can be opened, it is possible to beat the above lower bound. In the special case where only a constant number need to be opened, our protocol achieves a constant rate, which is optimal. Our protocols implements these restricted bit commitments from string commitments.

Trading robustness for correctness and privacy in certain multiparty computations, beyond an honest majority

In the second paper, we implement a tradeoff between robustness (everyone gets the right answer), privacy (each player's input remains private) and correctness (if a player gets an output, it is the correct one). We show that under the assumption of a simultaneous broadcast channel, we can trade privacy and correctness for robustness for a restricted class of functionalities which includes voting and anonymous message transmission. More precisely, for any threshold $t < n/2$ where n is the number of players, we can generate a protocol that is robust against t players and is private and correct against less than $n - t$ players. This trade-off is impossible for the general class of functionalities.

General Constructions of efficient multiparty computation

It is possible to generalize Oblivious Transfer. The sender can send any number of messages and the receiver is only allowed to learn certain subsets of those messages. We show how to construct Generalized Oblivious Transfer from a primitive called Verifiable Oblivious Transfer. We show how Generalized Oblivious Transfer can be used to implement important primitives that are traditionally based on very specific computations. We show how to implement Cut-and-Choose OT, modified Cut-and-Choose OT and Multi-Sender k-out-of-n OT. These primitives serve as an important backbone to secure computation protocols. We also show how Verifiable Oblivious Transfer can be constructed from bilinear pairings.

Efficient Secure Two-Party Computation From Black-Box Primitives via Faulty Cut-And-Choose OT

In the fourth paper, we focus on secure two-party computation. The paper builds upon the ideas of the third paper. The goal of this paper was to construct an efficient protocol that relies only on black-boxes for Oblivious Transfer, commitment schemes and garbled circuits. We took the construction of [LP11], replaced the cut-and-choose OT with a faulty cut-and-choose OT. We then had to integrate privacy amplifcation in their protocol to prevent leakage.

Personal contributions

On the directions of my supervisor, I was advised to try to construct efficient commitment protocols based on noisy channels. In [RTWW11], I contributed to the notion of restricted bit commitments and the applications of cover-free families.

Originally, Anne Broadbent, Stacey Jeffery and Alain Tapp had designed a protocol for a restricted class of functionalities which included voting that was private and correct against any subset of players. Unfortunately, any player could make the protocol abort. Based on their work, I was able to obtain a tradeoff of robustness for correctness and privacy. This was done by the introduction of the concepts of ghosts. I also proposed the anonymous message transmission protocol. Those results appeared in [BJRT12].

In General Constructions of efficient multiparty computation, I constructed Generalized

Oblivious Transfer from Verifiable Oblivious Transfer and then applied it to implement many of the oblivious primitives in the literature. This work will be submitted to ICITS 2014 [DNRT12].

It was due to the work on Generalized Oblivious Transfer, that I realised that efficient two-party computation could be constructed from a basic oblivious transfer by combining faulty Generalized Oblivious Transfer with privacy amplification. The protocol was later simplified by directly implementing a faulty version of the Cut-and-Choose OT from [LP11]. At the time this thesis was submitted, this work was in the process of being refereed by the CRYPTO 2014 committee [RT13].

CHAPTER 4

ON THE EFFICIENCY OF BIT COMMITMENT REDUCTIONS

Authors: Samuel Ranellucci, Alain Tapp, Severin Winkler, Jürg Wullschleger

4.1 Abstract

Two fundamental building blocks of secure two-party computation are *oblivious transfer* and *bit commitment*. While there exist unconditionally secure implementations of oblivious transfer from noisy correlations or channels that achieve constant rates, similar constructions are not known for bit commitment.

In this chapter, it will be demonstrated that any protocol that implements n instances of bit commitment with an error of at most 2^{-k} needs at least $\Omega(kn)$ instances of a given resource such as oblivious transfer or a noisy channel. This implies in particular that it is impossible to achieve a constant rate.

It will then be shown that it is possible to circumvent the above lower bound by restricting the way in which the bit commitments can be opened. In the special case where only a constant number of instances can be opened, the protocol achieves a constant rate, which is optimal. The protocol implements these restricted bit commitments from string commitments and is universally composable. The protocol provides significant speed-up over individual commitments in situations where restricted commitments are sufficient.

Keywords: secure two-party computation, bit commitment, string commitment, oblivious transfer, noisy channel, information theory

4.2 Introduction

Commitment schemes [Blu83] are one of the basic building blocks of two-party computation [Yao82]. Commitments can be used in *coin-flipping* [Blu83], *zero-knowledge proofs* [GMR85, GMW91], *zero-knowledge arguments* [BCC88] or as a tool in general two-party computation protocols to prevent malicious players from actively cheating (see for example [CvdGT95]).

A commitment scheme has two phases. In the *commit* phase, the sender has to decide on a value b. After the commit phase the value b is fixed and cannot be changed, while the receiver still does not get any information about its value. At a later time, the players may execute the second phase, called the *open* phase, where the bit b is revealed to the receiver. The scheme is called a *bit commitment* if b is only one bit, and it is called a *string commitment* if b is a longer bit string.

Efficiency of Reductions Bit commitments can be implemented from a wide variety of information-theoretic primitives [Cré97, DKS99, WNI03, Wul09]. There are protocols which implement a single string commitment from noisy channels at a constant rate, meaning that the size of the string grows linearly with the number of instances of noisy channels used, which is essentially optimal [WNI03]. Protocols which implement individual bit commitments at a constant rate, however, are not known. In [NOIMQ03] it has been shown that in any perfectly correct and perfectly hiding non-interactive bit commitment scheme from distributed randomness with a security of 2^{-k}, the size of the randomness given to the players must be at least $\Omega(k)$.

Another primitive which is of fundamental importance in two-party computation is oblivious transfer (OT) [Wie83, Rab81, EGL85]. Oblivious transfer can be implemented from noisy channels [Cré87, CK88, Cré97, CMW05], cryptogates [Kil00] and weak variants of noisy channels [DKS99, DFMS04, Wul07, Wul09]. While all these protocols require $\Omega(k)$ instances of a given primitive to implement a single OT with a security of 2^{-k}, it has been shown in [HIKN08, IPS08, IKOS09] that there are more efficient protocols if many OTs are implemented at once. In the semi-honest model and in some cases also in the malicious model, it is possible to implement OT *at a constant rate*, which means n instances of OT can be implemented from just $O(n)$ instances of the given primitive, if n is big enough compared to the security parameter. It is, therefore, possible to achieve the lower bound for such reductions [DM99, BM04, WW05, WW10] up to a constant factor. In the following, the question whether such efficient protocols also exist in the case of bit commitment is addressed.

4.2.1 Contribution

It will be shown that — in contrast to implementations of OT — no constant rate reduction can exist. More precisely, in Theorem 4 it will be shown that if a protocol implements n

bit commitments with a security of at least 2^{-k} from distributed randomness, then the mutual information between the sender's and the receiver's randomness must be almost kn or larger. This implies that at least $\Omega(kn)$ instances of oblivious transfer or noisy channels are needed to implement n bit commitments, and hence executing for each bit commitment a protocol that uses $O(k)$ instances is optimal. In combination with the lower bound from [WNI03], this bound can be generalized to string commitments: any protocol that implements n string commitments of length m needs at least $\Omega(n(k+m))$ bits of distributed randomness. This result is somewhat surprising, as OT is generally harder to implement than commitments.

However, in many applications of bit commitments the full properties of the commitment scheme is not required. For example in the famous zero-knowledge protocol of [GMW91], it is only required that a constant number of committed bits be opened. It will be shown that restricting the ways in which the bit commitments can be opened enables us to implement more efficient schemes that circumvent our impossibility result.[1] A new concept named *bit commitments with restricted openings* will be introduced. It allows a sender to commit to N bits, from which he may open up to $r < N$ one by one. After that, he may only open all the remaining bits at once. The protocol uses so-called *cover-free families*, and implements bit commitments with restricted openings from string commitments. Together with a simple construction of a cover-free family from [EFF85], the results imply that for any prime power q, $N = q^2$ bit can be implemented from which r can be opened from $(r+1)q$ string commitments of length q. (See Corollary 17 for the more general statement.) Together with the protocol from [WNI03], for any constant r, a constant-rate bit commitment protocol from noisy channels can be achieved. As bit commitments with restricted openings are strictly stronger than a string commitment, this is optimal. Together with another construction of a cover-free family from [DBV03], it is possible to implement $N = 2^{\Omega(n/r^2)}$ bit commitments from n string commitments. The security of protocols is proven in the Universal Composability model (UC) [Can01].

We will prove our lower bounds for independent bit commitments in Section 4.3. In Section 4.4, we introduce commitments with restricted openings and give reductions to string commitments. Note that Section 4.4 can be read without reading Section 4.3.

1. Note that for the specific case of zero-knowledge proofs other, more efficient, techniques are known [KMO89].

4.2.2 Notation

In the following, the probability distribution of a random variable X is denoted by $P_X(x)$ which defines the probability that X outputs x. The joint distribution $P_{XY}(x, y)$ implies a conditional distribution $P_{X|Y}(x, y)$, for all y with $P_Y(y) > 0$. The *statistical distance* between the distributions P_X and $P_{X'}$ over the domain $\mathcal{X}$ is defined as

$$\delta(P_X, P_{X'}) := \max_D \mid \Pr[D(X) = 1] - \Pr[D(X') = 1] \mid ,$$

where it is maximized over all (inefficient) distinguishers $D : \mathcal{X} \to \{0, 1\}$. The notation $[n]$ is used for the set $\{1, \ldots, n\}$. For a sequence $x = (x_1, \ldots, x_n)$ and $t \in [n]$, x^t denotes the subsequence $(x_1, \ldots, x_t)$.

4.2.3 Information Theory

The following tools from information theory will be used in the proof. It is assumed that the reader is familiar with the basic concepts of information theory, and refer to [CT12, HK02] for more details. The *conditional Shannon entropy* of X given Y is defined as [2]

$$\mathrm{H}(X \mid Y) := -\sum_{x,y} P_{XY}(x, y) \log P_{X|Y}(x, y) .$$

The following notation is used

$$h(p) = -p \log(p) - (1 - p) \log(1 - p)$$

for the binary entropy function, i.e., $h(p)$ is the entropy of the Bernoulli distribution [3] with parameter p. The *mutual information* of X and Y given Z is defined as

$$\mathrm{I}(X; Y \mid Z) = \mathrm{H}(X \mid Z) - \mathrm{H}(X \mid YZ) .$$

2. All logarithms are binary, and use the convention that $0 \cdot \log 0 = 0$.

3. The Bernoulli distribution with parameter $p \in [0, 1]$ takes on the value 1 with probability p and 0 otherwise.

The mutual information satisfies the following chain rule

$$\mathrm{I}(X_1, \ldots, X_n; Y) = \sum_{i=1}^{n} \mathrm{I}(X_i; Y \mid X_1, \ldots, X_{i-1}).$$

The Kullback-Leibler divergence or relative entropy of two distributions P_X and Q_X on $\mathcal{X}$ is defined as

$$\mathrm{D}(P_X \parallel Q_X) = \sum_{x \in \mathcal{X}} P_X(x) \log \frac{P_X(x)}{Q_X(x)} \, .$$

The conditional divergence of two distributions P_{XY} and Q_{XY} on $\mathcal{X} \times \mathcal{Y}$ is defined as

$$\mathrm{D}(P_{Y|X} \parallel Q_{Y|X}) = \sum_{x \in \mathcal{X}} P_X(x) \, \mathrm{D}(P_{Y|X=x} \parallel Q_{Y|X=x}) \, .$$

The binary divergence of two probabilities p and q is defined as the divergence of the Bernoulli distributions with parameters p and q, i.e.,

$$\mathrm{d}(p \parallel q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q} \, .$$

The divergence (and hence also the conditional divergence) is always non-negative. Furthermore, we have the following chain rule

$$\mathrm{D}(P_{XY} \parallel Q_{XY}) = \mathrm{D}(P_X \parallel Q_X) + \mathrm{D}(P_{Y|X} \parallel Q_{Y|X}) \, . \tag{4.1}$$

This implies

$$\mathrm{D}(P_X P_{Y|X} \parallel P_X Q_{Y|X}) = \mathrm{D}(P_{Y|X} \parallel Q_{Y|X}). \tag{4.2}$$

Let Q_X and P_X be two distributions over the inputs to the same channel $P_{Y|X}$. Then the divergence between the outputs $P_Y = \sum_x P_X P_{Y|X}$ and $Q_Y = \sum_x Q_X P_{Y|X}$ of the channel is not greater than the divergence between the inputs, i.e., the divergence satisfies the data-processing inequality

$$\mathrm{D}(P_X \parallel Q_X) \geq \mathrm{D}(P_Y \parallel Q_Y) \, . \tag{4.3}$$

Furthermore, for random variables X, Y and Z distributed according to P_{XYZ}

$$\mathrm{I}(X;Y \mid Z) = \mathrm{D}(P_{X|YZ} \parallel P_{X|Z}) \ . \tag{4.4}$$

Let $P_{X|Y=y} = P_{X|Y=y,Z=z}$ for all y, z (or $P_{Z|Y=y} = P_{Z|Y=y,X=x}$ for all y, z, which is equivalent). Then we say that X, Y and Z form a Markov-chain, denoted by $X \leftrightarrow Y \leftrightarrow Z$. If $W \leftrightarrow XZ \leftrightarrow Y$, then

$$I(X;Y \mid ZW) \leq I(X;Y \mid Z) \ . \tag{4.5}$$

4.3 Impossibility Results

4.3.1 Model and Security Definition

The following model will be considered: a trusted third party chooses random variables (U, V) according to a distribution P_{UV} and sends U to the sender and V to the receiver. The sender receives an input bit $b \in \{0, 1\}$. In the commit phase, the players exchange messages in several rounds. Let all the messages exchanged be M, which is a randomized function of (U, V, b). In the open phase, the sender sends b together with a value D_1 to the receiver. The receiver then sends a message E_1 to the receiver, who replies with a message D_2 and so on. Let $N := (D_1, E_1, D_2, E_2, \ldots, E_{t-1}, D_t)$ be the total communication in the open phase. (It is assumed that the number of rounds in the open phase is upper bounded by a constant t. By padding the protocol with empty rounds it can thus be assumed without loss of generality that the protocol uses t rounds in every execution.) Finally, the receiver accepts or rejects, which is modeled by a randomized function $F(b, V, M, N)$ that outputs 1 for accept and 0 for reject. Let the distribution in the honest setting be $P_{UVMN|B=b}$. Three parameters are defined that quantify the security for the sender and the receiver, respectively, and the correctness of the protocol.

— ε-*correct*: $\Pr[F(b, V, M, N) = 1] \geq 1 - \varepsilon$.

— β-*hiding*: $\delta(P_{VM|B=0}, P_{VM|B=1}) \leq \beta$.

— γ-*binding*: For any $b \in \{0, 1\}$ and for any malicious sender that is honest in the commit phase on input b and tries to open $1 - b$, we have $\Pr[F(1 - b, V, M, N') = 1] \leq \gamma$, where N' is the communication between the malicious sender and the honest receiver

in the open phase.

Note that the above security conditions are not sufficient to prove the security of a protocol [4], but any sensible security definition for commitments implies these conditions. Since we only use the definition to prove the non-existence of certain protocols, this makes the result stronger.

4.3.2 Lower Bound for Multiple Bit Commitments

In the following we prove a new lower bound on the mutual information between the randomness of the sender and the randomness of the receiver in any bit commitment protocol. First, we show the following technical lemma.

Lemma 1 *If a protocol that implements bit commitment from randomness (U, V) is γ-binding, ε-correct and β-hiding, then*

$$\mathrm{d}(1-\varepsilon \parallel \gamma+\beta) \leq \sum_{i=1}^{t} \mathrm{I}(D_i; V \mid MD^{i-1}E^{i-1}). \tag{4.6}$$

where M is the whole communication in the commit phase and $N = (D, E) = (D_1, E_1, \ldots, D_t)$ is the whole communication in the open phase.

Proof. Let $b \in \{0, 1\}$ and $\bar{b} := 1 - b$. Assume that the sender in the commit phase honestly commits to b. If she honestly opens b in the open phase, the communication can be modeled by a channel $P_{DE|VM}$ (that may depend on b) and the resulting distribution is

$$P_{DEVM|B=b} = P_{DE|VM} P_{VM|B=b} \ ,$$

We have omitted U as it does not play a role in the following arguments. The correctness property implies that an honest receiver accepts values drawn from this distribution with probability at least $1 - \varepsilon$. Let the sender commit to $\bar{b}$ and then try to open b by sampling her messages according to the distributions $P_{D_1|M}$ and $P_{D_i|MD^{i-1}E^{i-1}}$ for $2 \leq i \leq t$. (Note that the sender does not know V and, therefore, chooses her messages independently of V.)

4. To prove the security of a protocol one has to consider for example a malicious sender in the commit phase.

The communication in the opening phase can be modeled by a channel

$$Q_{DE|VM} := P_{D_1|M} P_{E_1|VMD_1} \dots P_{D_t|MD^{t-1}E^{t-1}} \ .$$

The binding property implies that the receiver accepts values distributed according to $P_{VM|B=\bar{b}} Q_{DE|VM}$ with probability at most γ. $\delta(P_{VM|B=b}, P_{VM|B=\bar{b}}) \leq \beta$ implies that

$$\delta(P_{VM|B=b} Q_{DE|VM}, P_{VM|B=\bar{b}} Q_{DE|VM}) \leq \beta,$$

and hence values drawn from the distribution $P_{VM|B=b} Q_{DE|VM}$ are accepted with probability at most $\gamma + \beta$. Note that the bit indicating acceptance can also be modeled by a channel $P_{F|DEVM}$. Thus, we can apply the data-processing inequality (4.3) to bound $\mathrm{d}(1-\varepsilon \parallel \gamma+\beta)$. Using the chain rule (4.1) and the non-negativity of the relative entropy, we have (we omit conditioning on $B = b$ in the following)

$$\begin{aligned}
\mathrm{d}(1-\varepsilon \parallel \gamma+\beta) &\leq \mathrm{D}(P_{VM} P_{DE|VM} \parallel P_{VM} Q_{DE|VM}) \\
&= \mathrm{D}(P_{DE|VM} \parallel Q_{DE|VM}) \\
&= \sum_{i=1}^{t} \mathrm{D}(P_{D_i|VMD^{i-1}E^{i-1}} \parallel P_{D_i|MD^{i-1}E^{i-1}}) \\
&\quad + \sum_{i=1}^{t-1} \mathrm{D}(P_{E_i|VMD^iE^{i-1}} \parallel P_{E_i|VMD^iE^{i-1}}) \\
&= \sum_{i=1}^{t} \mathrm{D}(P_{D_i|VMD^{i-1}E^{i-1}} \parallel P_{D_i|MD^{i-1}E^{i-1}}) \\
&= \sum_{i=1}^{t} \mathrm{I}(D_i; V \mid MD^{i-1}E^{i-1})
\end{aligned}$$

□

The following lemma from [OW05] gives a lower bound on the binary divergence, which we will use to bound the right-hand side of (4.6) in the following.

Lemma 2 (Lower Bound on the binary divergence)

$$\mathrm{d}(a \parallel b) \geq \frac{(a-b)^2}{1-2b} \log \frac{1-b}{b}$$

Let $\varepsilon = \beta = \gamma = 2^{-k}$. Then, for $k \geq 3$, we have

$$\begin{aligned} \mathrm{d}(1-\varepsilon \parallel \gamma+\beta) &= \frac{(1-3\cdot 2^{-k})^2}{1-2^{-k+2}} \log(2^{k-1}-1) \\ &\geq (k-2)\cdot \frac{1-2^{-k+3}}{1-2^{-k+2}} \geq (k-2)\cdot \frac{2^{k-2}-2}{2^{k-2}-1}. \end{aligned} \tag{4.7}$$

Next, we prove a lower bound on the information that the communication in the open phase must reveal about the receiver's randomness V for any protocol that implements bit commitment from a shared distribution P_{UV}. This lower bound is essentially k if the error of the protocol is at most 2^{-k}.

Lemma 3 *Let $k \geq 3$. Then any 2^{-k}-secure bit commitment must have*

$$\mathrm{I}(N;V \mid M) - \mathrm{I}(N;V \mid UM) = \mathrm{I}(U;V \mid M) - \mathrm{I}(U;V \mid MN) \geq (k-2)\cdot \frac{2^{k-2}-2}{2^{k-2}-1}\ .$$

where M and N is the whole communication in the commit phase and in the open phase, respectively.

Proof. Consider a protocol over t rounds in the open phase, i.e., the whole communication is $N = (D, E) = (D_1, E_1, \ldots, D_t)$.

Furthermore, from $E_i \leftrightarrow VMD^iE^{i-1} \leftrightarrow U$ and inequality (4.5) follows that for all i

$$\mathrm{I}(E_i;V \mid MD^iE^{i-1}) \geq \mathrm{I}(E_i;V \mid UMD^iE^{i-1})\ .$$

Hence, we have

$$\begin{aligned} \mathrm{I}(N;V \mid M) &= \sum_i \mathrm{I}(E_i;V \mid MD^iE^{i-1}) + \sum_i \mathrm{I}(D_i;V \mid MD^{i-1}E^{i-1}) \\ &\geq \sum_i \mathrm{I}(E_i;V \mid UMD^iE^{i-1}) + \sum_i \mathrm{I}(D_i;V \mid MD^{i-1}E^{i-1}) \end{aligned}$$

and

$$\begin{aligned}\mathrm{I}(U;V \mid MN) &= \mathrm{I}(NU;V \mid M) - \mathrm{I}(N;V \mid M) \\ &= \mathrm{I}(U;V \mid M) + \sum_i \mathrm{I}(E_i;V \mid UMD^iE^{i-1}) - \mathrm{I}(N;V \mid M) \\ &\leq \mathrm{I}(U;V \mid M) - \sum_i \mathrm{I}(D_i;V \mid MD^{i-1}E^{i-1}) \ .\end{aligned}$$

The statement now follows from Lemma 1 and inequality (4.7).

□

Next, we consider implementations of n individual bit commitments. The sender gets input $b^n = (b_1, \ldots, b_n)$ and commits to all bits at the same time, which results in the overall distribution

$$P_{UVM|B^n=b^n} = P_{UV} P_{M|UV,B^n=b^n} \ .$$

after the commit phase. To reveal the ith bit, the sender and the receiver interact resulting in the transcript N_i. The following theorem says that the mutual information between the sender's randomness U and the receiver's randomness V must be almost kn to implement n bit commitments with an error of at most 2^{-k}. The proof uses Lemma 3 to lower bound the information that the sender must reveal about V for every bit that he opens.

Theorem 4 *Let $k \geq 3$. Then any 2^{-k}-secure protocol that implements n bit commitments from randomness (U, V) must have for all $b^n \in \{0,1\}^n$*

$$\mathrm{I}(U;V) \geq \mathrm{I}(U;V \mid M, B = b^n) \geq n(k-2) \cdot \frac{2^{k-2} - 2}{2^{k-2} - 1} \ .$$

Proof. Let $\hat{i} \in [n]$. We first construct a commitment to a single bit, which will allow us to apply the bound from Lemma 3. This bit commitment is defined as follows: to commit to the bit b, the players execute the commit phase on input b^n, which is equal to the input bit b on position $\hat{i}$ and equal to the constant $\hat{b}^n$ on all other positions. Additionally, (still as part of the commit phase), the sender opens the first $\hat{i} - 1$ commitments, which means that the messages $N^{\hat{i}-1}$ get exchanged. To open the commitment, the sender opens bit $\hat{i}$. This bit commitment scheme has at least the same security as the original commitment. Thus,

Lemma 3 implies that

$$\mathrm{I}(U;V \mid MN^{\hat{i}}) \leq \mathrm{I}(U;V \mid MN^{\hat{i}-1}) - (k-2) \cdot \frac{2^{k-2}-2}{2^{k-2}-1}. \tag{4.8}$$

Since this holds for all $\hat{i}$, we can apply (4.8) repeatedly to get

$$\begin{aligned} 0 &\leq \mathrm{I}(U;V \mid MN^{n}) \\ &\leq \mathrm{I}(U;V \mid MN^{n-1}) - (k-2) \cdot \frac{2^{k-2}-2}{2^{k-2}-1} \\ &\leq \mathrm{I}(U;V \mid M) - n(k-2) \cdot \frac{2^{k-2}-2}{2^{k-2}-1} \end{aligned}$$

By induction over all rounds of the commit protocol using (4.5) (see, for example, [WW10] for a detailed proof) it follows that

$$I(U;V \mid M) \leq I(U;V)\ .$$

This implies the statement.

□

It is possible to securely implement 1-out-of-2 bit oblivious transfer $\binom{2}{1}$-OT^1 from randomness distributed according to P_{UV} with $\mathrm{I}(U;V) = 1$ [BBCS92, Bea96]. A binary symmetric noisy channel ((p)-BSNC) with crossover probability p can be implemented from randomness distributed according to P_{UV} with $\mathrm{I}(U;V) = 1 - h(p)$. Together with these reductions, Theorem 4 implies that (almost) kn instances of $\binom{2}{1}$-OT^1 or $kn/(1-h(p))$ instances of a (p)-BSNC are needed to implement n bit commitments with an error of at most 2^{-k}.

There exists a universally composable protocol [5] that implements bit commitment from $2k$ instances of $\binom{2}{1}$-OT^1 with an error of at most 2^{-k}. Thus, n bit commitments can be implemented from $2n(k+\log(n))$ instances of $\binom{2}{1}$-OT^1 with an error of at most $n \cdot 2^{-(k+\log(n))} = 2^{-k}$ using n parallel instances of this protocol. Theorem 4 shows that this is optimal up to a factor of 4 if $k \geq \log(n)$.

5. See for example Claim 33 in the full version of [Can01].

4.3.3 Lower Bounds for Multiple String Commitments

A *string commitment* is a generalization of bit commitment where the sender may commit to a string of length $\ell \geq 1$. It is weaker than ℓ instances of bit commitment because the sender has to reveal all bits simultaneously. In [WNI03] a lower bound on the conditional entropy of the sender's randomness U given the receiver's randomness V for any string commitment protocol from randomness (U, V) has been shown. This bound essentially says that $\mathrm{H}(U \mid V)$ must be greater than or equal to ℓ to implement a string commitment of length ℓ. The following lemma provides a similar bound for the security definition considered here. (The proof is given in the appendix.)

Lemma 5 *If any protocol implements an ℓ-bit string commitment from randomness (U, V) is ε-correct, $\beta-$hiding and γ-binding, then*

$$\mathrm{H}(U \mid V) \geq (1 - \varepsilon - \beta - \gamma)\ell - h(\beta) - h(\varepsilon + \gamma).$$

Together with the bound of Theorem 4, we obtain the following lower bound on the randomness of the sender in any bit commitment protocol.

Corollary 6 *Let $k \geq 3$. For any protocol that implements n individual m-bit string commitments from randomness (U, V) with an error of at most 2^{-k}*

$$\mathrm{H}(U) \geq n(k + \ell - 2) \cdot \frac{2^{k-2} - 2}{2^{k-2} - 1} - 3 \cdot 2^{-k} \cdot n\ell - 3h(2^{-k}).$$

Proof. Using Lemma 5 and Theorem 4, we get

$$\begin{aligned}
\mathrm{H}(U) &= \mathrm{I}(U; V) + \mathrm{H}(U \mid V) \\
&\geq n(k-2) \cdot \frac{2^{k-2} - 2}{2^{k-2} - 1} + (1 - 3 \cdot 2^{-k})n\ell - h(2^{-k}) - h(2^{-k+1}) \\
&\geq n(k + \ell - 2) \cdot \frac{2^{k-2} - 2}{2^{k-2} - 1} - 3 \cdot 2^{-k} \cdot n\ell - 3h(2^{-k}).
\end{aligned}$$

□

In [BMSW02] it has been shown that any non-interactive perfectly hiding and perfectly correct bit commitment protocol from distributed randomness P_{UV} is at most $(2^{-H(V|U)})$-

binding. This result implies stronger bounds than Theorem 4 and Lemma 5 for certain reductions. The following lemma provides a lower bound on the uncertainty of the sender about the receiver's randomness for any bit commitment protocol. This lower bound is essentially equal to k if the protocol is 2^{-k}-secure and implies, in particular, the result from [BMSW02].

Lemma 7 *If a protocol that implements bit commitment from randomness (U, V) is γ-binding, ε-correct and β-hiding, then*

$$\mathrm{d}(1-\beta-\varepsilon \parallel \gamma) \leq \mathrm{H}(V \mid UM) \leq \mathrm{H}(V \mid U).$$

where M is the whole communication in the commit phase. If $\beta = \gamma = \varepsilon = 2^{-k}$, then

$$\mathrm{H}(V \mid U) \geq (k-1) \cdot \frac{2^{k-1}-4}{2^{k-1}-1}. \tag{4.9}$$

Proof. We have $\delta(P_{VM|B=b}, P_{VM|B=\bar{b}}) \leq \beta$. This implies that the distribution $P_{U|VM,B=\bar{b}} P_{VM|B=b}$ is β-close to $P_{UVM|B=\bar{b}}$. Thus, when the sender honestly opens $\bar{b}$ starting from values distributed according $P_{U|VM,B=\bar{b}} P_{VM|B=b}$, the receiver accepts the resulting values with probability at least $1-\beta-\varepsilon$. We consider the following attack: the sender honestly commits to b, generates v' by applying $P_{V|UM,B=b}$ and then generates u by applying the channel $P_{U|VM,B=\bar{b}}$ to (v', m). When the sender now tries to open $\bar{b}$, the binding property guarantees that the receiver accepts the resulting values with probability at most γ. Thus, we can apply the data-processing inequality (4.3) to bound $\mathrm{d}(1-\beta-\varepsilon \parallel \gamma)$. Let V' be a copy of V, i.e., a random variable with distribution $P_{VV'}(v, v) = P_V(v)$. Using the chain rule (4.2), we have

$$\begin{aligned}
\mathrm{d}(1-\beta-\varepsilon \parallel \gamma) &\leq \mathrm{D}(P_{VV'|UM,B=b} P_{UM|B=b} \parallel P_{V|UM,B=b} P_{V|UM,B=b} P_{UM|B=b}) \\
&\leq \mathrm{D}(P_{VV'|UM,B=b} \parallel P_{V|UM,B=b} P_{V|UM,B=b}) \\
&= \mathrm{H}(V \mid UM, B=b) \\
&\leq \mathrm{H}(V \mid U).
\end{aligned}$$

Using Lemma 2 this implies inequality (4.9).

□

Consider a protocol that implements bit commitment with security of 2^{-k} from N instances of $\binom{2}{1}$-OT^M where M denotes the length of the string transfered. Since $\binom{2}{1}$-OT^M can be reduced to a shared distribution P_{UV} with $H(V|U) = 1$, Lemma 7 implies that $N \geq (k-2) \cdot \frac{2^{k-1}-4}{2^{k-1}-1}$, i.e., one needs, independently of M, almost k instances of OT.

Together with Theorem 4 and Lemma 5, this implies the following lower bound on the number of instances of OT needed to implement multiple string commitments, which demonstrates that all three lower bounds can be meaningful in this scenario.

Corollary 8 *Let $k \geq 3$. For any protocol that implements n individual ℓ-bit string commitments with an error of at most 2^{-k} from N instances of $\binom{2}{1}$-OT^M*

$$N \geq \max\left(\frac{(1-3\cdot 2^{-k})\ell n - 3h(2^{-k})}{M}, (k-2)\cdot\frac{n}{M}\cdot\frac{2^{k-2}-2}{2^{k-2}-1}, (k-1)\cdot\frac{2^{k-1}-4}{2^{k-1}-1}\right).$$

4.4 Commitments with restricted openings

In this section, we will present protocols that implement commitments with restricted openings from several instances of string commitment. We will use the Universal Composability model [Can01], and assume that the reader is familiar with it. In our proof, we will only consider static adversaries. For simplicity, we omit session IDs and players IDs.

String Commitment is a functionality that allows the sender to commit to a string of n bits, and to reveal the whole string later to the receiver. The receiver does not get to know the string before it is opened, and the sender cannot change the string once he has sent it.

Definition 9 (String-Commitment) The functionality $\mathcal{F}^n_{\mathtt{SCOM}}$ behaves as follows:

— Upon input (`commit`, b) with $b \in \{0,1\}^n$ from the sender: check that commit has not been sent yet. If so, send `committed` to the receiver and store b. Otherwise, ignore the message.

— Upon input `openall` from the sender: check if there has been a commit message before, and the commitment has not been opened yet. If so, send (`openall`, b) to the receiver and ignore the message otherwise.

Note that given $\mathcal{F}^n_{\mathtt{SCOM}}$, it is possible to commit to individual bits at different times: the sender simply commits to a random string $b' = (b'_1, \ldots, b'_n)$, and whenever he wants to commit

to a bit b_i for $i \in [n]$, he sends $b_i \oplus b'_i$ to the receiver. On the other hand, it is not possible to open bits at different times using $\mathcal{F}_{\texttt{SCOM}}^n$.

Bit commitment is a string commitment of length 1, i.e., $\mathcal{F}_{\texttt{BCOM}} := \mathcal{F}_{\texttt{SCOM}}^1$. We denote n independent bit commitments by $(\mathcal{F}_{\texttt{BCOM}})^n$. Since $(\mathcal{F}_{\texttt{BCOM}})^n$ does allow bits to be opened at different times, it is strictly stronger than $\mathcal{F}_{\texttt{SCOM}}^n$. However, as we have seen in the last section, $(\mathcal{F}_{\texttt{BCOM}})^n$ can is also quite expensive, in terms of resources needed, to implement. Therefore, we define a primitive that is somewhere between these two: *commitments with restricted openings* allow a sender to commit to n bits, but then he may only open r individual bits of his choice one by one. To open more than r bits, he has to open the remaining bits all together.

Definition 10 (Commitments with restricted openings) The functionality $\mathcal{F}_{\texttt{RCOM}}^{n,r}$ behaves as follows:

— Upon input (`commit`, b) with $b \in \{0,1\}^n$ from the sender: check that commit has not been sent yet. If so, send `committed` to the receiver and store b. Otherwise, ignore the message.

— Upon input (`open`,i) with $i \in [n]$ from the sender: check that there has been a commit message before, and that i has not been opened yet. Also check that the number of opened values so far is smaller than r. If so, send (`open`, i, b_i) to the receiver and ignore the message otherwise.

— Upon input `openall` from the sender: check if there has been a commit message before, and no `openall` message has been received yet from the sender. If so, send (`openall`, b) to the receiver and ignore the message otherwise.

For $r = 0$ and $r = n$, commitment with restricted openings are equivalent to string commitments and individual bit commitments, respectively: $\mathcal{F}_{\texttt{SCOM}}^n \equiv \mathcal{F}_{\texttt{RCOM}}^{n,0}$ and $(\mathcal{F}_{\texttt{BCOM}})^n \equiv \mathcal{F}_{\texttt{RCOM}}^{n,n}$.

4.4.1 Warm-up: A Protocol for $r = 1$

Let $n \geq s \geq 1$. We define $N := \binom{n}{s}$. Let $f(i)$ be a bijection from $[N]$ to the subsets of $[n]$ of size s. We now present a protocol that implements $\mathcal{F}_{\texttt{RCOM}}^{N,1}$ from n instances of $\mathcal{F}_{\texttt{SCOM}}^N$. We denote the ith bit of string c_j by $c_{j,i}$.

Protocol 1:

— When the sender receives (`commit`, b), he chooses n uniformly chosen strings $c_1, \ldots c_n \in \{0,1\}^N$, with the restriction that for all $i \in [N]$ $\oplus_{j \in f(i)} c_{j,i} = b_i$. For all $j \in [n]$, the sender sends (`commit`, c_j) to the jth instance of $\mathcal{F}_{\texttt{SCOM}}^N$. After that he ignores all messages (`commit`, b').

— When the receiver has received `committed` from all instances of $\mathcal{F}_{\texttt{SCOM}}^N$, he outputs `committed`.

— When the sender receives `(open,`i`)`, he sends `(open,`i`)` to the receiver and `openall` to all instances of $\mathcal{F}_{\texttt{SCOM}}^N$ in $f(i)$. After that, he ignores all messages `(open,`i'`)`, for any i'.

— When the receiver receives `(open,`i`)` from the sender, he waits until for all $j \in f(i)$, he has received a message (`open`, c_j) from the jth instance of $\mathcal{F}_{\texttt{SCOM}}^N$. He then outputs (`open`, i, $\oplus_{j \in f(i)} c_{j,i}$). After that, he ignores all messages `(open,`i'`)` for any i'.

— When the sender receives `openall`, he sends `openall` to the receiver and to all instances of $\mathcal{F}_{\texttt{SCOM}}^N$. After that, he ignores all messages `openall`.

— When the receiver receives `openall` from the sender, he waits until for all $j \in [n]$ he has received a message (`open`, c_j) from the jth instance of $\mathcal{F}_{\texttt{SCOM}}^N$. He then outputs (`openall`, b'), where $b'_i := \oplus_{j \in f(i)} c_{j,i}$, for all $i \in [N]$. After that, he ignores all messages `openall`.

Theorem 11 *For any $n \geq s \geq 1$ and $N = \binom{n}{s}$ Protocol 1 UC-implements $\mathcal{F}_{RCOM}^{N,1}$ from $\left(\mathcal{F}_{SCOM}^N\right)^n$.*

Since Protocol 1 is a special case of Protocol 2, Theorem 11 follows from Theorem 15. We will, therefore, only provide a proof sketch and refer to the proof of Theorem 15 for details.

Proof. [Proof Sketch] It is easy to verify that the protocol is correct if the two players are honest. Furthermore, since the opened values are calculated from the values the sender is committed to, it is also clear that the protocol must be binding. That the protocol is hiding before anything is opened is obvious. It remains to show that the protocol is hiding after one bit has been opened. Since one subset of size k cannot be a proper subset of any other subset of size k, all bit that have not been opened yet are shared among string commitments

from which at least one has not been opened. Hence, they remain perfectly hidden from the receiver.

□

Note that in each of the n instances of $\mathcal{F}_{\texttt{SCOM}}^{N}$ in Protocol 1, there are exactly $(n-s)/n \cdot N$ bits that are not used. Since they are at fixed positions and both players know where they are, they can be removed without changing the properties of the protocol. So the length of the string commitments used can be reduced to Ns/n.

Our protocol makes use of *cover-free families* [KS64a, EFF85, SWZ00, DBV03], which are a generalization of *Sperner sets* [Spe28] , which means that no element of the set is covered by any other. By strengthening this requirement to so-called *cover-free families*, Protocol 1 can be generalized in a natural way such that $r > 1$ bits can be opened. Cover-free families are also known as *superimposed codes* from [KS64b] and require that no set is covered by the union of r other sets.

Definition 12 Let $\mathcal{X}$ be a set of n elements and let $\mathcal{B}$ be a set of subsets of $\mathcal{X}$, then $(\mathcal{X}, \mathcal{B})$ is a r-cover-free family $r-\text{CFF}(\mathcal{X}, \mathcal{B})$ if for any r sets $B_{i_1}, \ldots B_{i_r} \in \mathcal{B}$, and any other $B \in \mathcal{B}$, it holds that

$$B \not\subseteq \bigcup_{j=1}^{r} B_{i_j} \ .$$

Example 13 All subsets of $[n]$ of size k form a cover-free family for $r = 1$, because there is no subset that completely covers any other subset.

Here is a simple example of a cover-free family for $r > 1$ given in [EFF85].

Example 14 [EFF85] Let q be a prime power, and $d, r \in \mathbb{N}$ such that $rd < q$. Let $\mathcal{X} = \mathcal{Y} \times \text{GF}(q)$, where $\mathcal{Y} \subseteq \text{GF}(q)$ and $|\mathcal{Y}| = rd+1$. An element B in the family $\mathcal{B}$ is constructed from a polynomial $p(y) := a_0 + y \cdot a_1 + ... + y^d \cdot a_d$ of degree d where $a_i \in \text{GF}(q)$ by $B := \{(y, p(y)) : y \in \mathcal{Y})\}$. Two polynomials of degree d intersect at most d times. Therefore, any union of r elements $B_1, \ldots B_r$ intersects any other element B at most $rd < |\mathcal{Y}|$ times, and therefore cannot cover B. $(\mathcal{X}, \mathcal{B})$ is therefore a r-cover-free family with $|\mathcal{X}| = (rd+1)q$ and $|\mathcal{B}| = q^{d+1}$.

We now give a protocol that implements $\mathcal{F}_{\texttt{RCOM}}^{N,r}$ from n instances of $\mathcal{F}_{\texttt{SCOM}}^{N}$ using a $r-\text{CFF}(\mathcal{X}, \mathcal{B})$, where $\mathcal{X} = \{1, \ldots, n\}$ and $\mathcal{B} = \{B_1, B_2, ..., B_N\}$.

Protocol 1:

— When the sender receives (`commit`, b), he chooses n uniformly chosen strings $c_1, \ldots c_n \in \{0,1\}^N$, with the restriction that for all $i \in [N]$ we have

$$\bigoplus_{j \in B_i} c_{j,i} = b_i \ .$$

For $j \in [n]$, the sender sends (`commit`, c_j) to the jth instances of $\mathcal{F}_{\texttt{SCOM}}^N$. After that he ignores all messages (`commit`, b').

— When the receiver has received `committed` from all instances of $\mathcal{F}_{\texttt{SCOM}}^N$, he outputs `committed`.

— For the first r times when the sender receives `(open,`i`)`, he sends `(open,`i`)` to the receiver and `openall` to all instances of $\mathcal{F}_{\texttt{SCOM}}^N$ in B_i, if they have not been opened yet. After that, he ignores all messages `(open,`i`)`.

— For the first r times when the receiver receives `(open,`i`)` from the sender and (`open`, c_j) from all instances $\mathcal{F}_{\texttt{SCOM}}^N$ in B_i, he outputs (`open`, $\bigoplus_{j \in B_i} c_{j,i}$). After that, he ignores these messages.

— When the sender receives `openall`, he sends `openall` to the receiver and to all instances of $\mathcal{F}_{\texttt{SCOM}}^N$. After that, he ignores all `openall` messages.

— When the receiver receives `openall` from the sender and (`open`, c_j) from all instances of $\mathcal{F}_{\texttt{SCOM}}^N$, he outputs (`openall`, $(b'_1, \ldots, b'_N)$), where $b'_i := \oplus_{j \in B_i} c_{j,i}$. After that, he ignores all messages `openall`.

Theorem 15 *Given an* $r-\mathrm{CFF}(\mathcal{X}, \mathcal{B})$ *where* $|\mathcal{X}| = n$ *and* $|\mathcal{B}| = N$*, protocol* 1 *UC-implements* $\mathcal{F}_{\texttt{RCOM}}^{N,r}$ *from* n *instances of* $\mathcal{F}_{\texttt{SCOM}}^N$*.*

Proof. It is easy to verify that the protocol is correct if the two players are honest.

Corrupted sender. First, we consider the case where the comitter is corrupted. He may send messages (`commit`, c_j) or `openall` to the instances of $\mathcal{F}_{\texttt{SCOM}}^N$, and message `(open,`$i$`)` or `openall` to the receiver.

Our simulator simulates the adversary, and records all messages sent out by the adversary. After receiving all messages (`commit`, c_j) to the instances of $\mathcal{F}_{\texttt{SCOM}}^N$, he calculates $b_i := \oplus_{j \in B_i} c_{j,i}$

for all i and sends (`commit`, $(b_1, \ldots, b_N)$) to $\mathcal{F}_{\text{RCOM}}^{N,r}$. After receiving `(open,`$i$`)` and all messages `openall` sent to the instances of $\mathcal{F}_{\text{SCOM}}^{N}$ in B_i, he sends `(open,`i`)` to $\mathcal{F}_{\text{RCOM}}^{N,r}$. After receiving `openall` sent to the receiver and all instances $\mathcal{F}_{\text{SCOM}}^{N}$, he sends `openall` to $\mathcal{F}_{\text{RCOM}}^{N,r}$. It is not difficult to verify that our simulation is perfect, and we get REAL $\equiv$ IDEAL.

Corrupted receiver. Let the receiver be corrupted by the adversary. He receives `committed` and (`open`,c_j) messages from the instances of $\mathcal{F}_{\text{SCOM}}^{N}$, and messages `(open,`$i$`)` and `openall` from the sender.

Our simulator simulates the adversary, and interacts with $\mathcal{F}_{\text{RCOM}}^{N,r}$ and the adversary. After receiving the `committed` message from $\mathcal{F}_{\text{RCOM}}^{N,r}$, it sends `committed` from all $\mathcal{F}_{\text{SCOM}}^{N}$ to the adversary. After receiving message `(open,`i,b_i`)` from $\mathcal{F}_{\text{RCOM}}^{N,r}$, he first sends `(open,`$i$`)` to the adversary. Then for all instances of $\mathcal{F}_{\text{SCOM}}^{N}$ in B_i which have not been opened yet, he chooses strings c_j uniformly at random, with the restriction that $\oplus_{j\in B_i} c_{j,i} = b_i$, and sends `(open,`$c_j$`)` from the jth instance of $\mathcal{F}_{\text{SCOM}}^{N}$ to the adversary. After receiving message `(openall,`b`)` from $\mathcal{F}_{\text{RCOM}}^{N,r}$, he first sends `openall` to the adversary. Then for all instances of $\mathcal{F}_{\text{SCOM}}^{N}$ which have not been opened yet, he chooses the strings c_j uniformly at random, with the restriction that $\oplus_{j\in B_i} c_{j,i} = b_i$, and sends `(open,`$c_j$`)` from the jth instance of $\mathcal{F}_{\text{SCOM}}^{N}$ to the adversary.

To show that this simulation in the ideal setting is identical to the real setting, we have to show that they are identical after each step. It is easy to see that this is the case before anything has been opened, and after `openall` has been executed.

$\mathcal{F}_{\text{RCOM}}^{N,r}$ allows the sender to open at most r values. Assume that $s \leq r$ have been opened so far. Since $\mathcal{B}$ is a $r-\text{CFF}(\mathcal{X}, \mathcal{B})$, there is at least one instance of $\mathcal{F}_{\text{SCOM}}^{N}$ in B_i for all the remaining $i \in [N]$ that has not been opened yet. Since the ith bit of that string is uniform and all the ith bits of the strings in B_i add up to b_i, the bits at the ith position of all the opened strings are uniform and independent of each other and of the bit b_i. Therefore, the simulated values c_j sent to the adversary have the same distribution in the real and in the ideal setting. The simulation is again perfect, and we get REAL$\equiv$ IDEAL.

□

Corollary 16 *For any $n \geq s \geq 1$ and $N = \binom{n}{s}$ there exists a protocol that UC-implements $\mathcal{F}_{RCOM}^{N,1}$ from $\left(\mathcal{F}_{SCOM}^{Ns/n}\right)^n$.*

If we use the cover-free family of Example 14, then the size can be reduced by a factor of q because we can let all the bit commitments which have different values a_0 but the same

values $a_1, ..., a_d$ share the same position in the string commitments. For $s = 2$, we have $N = n(n-1)/2$. So from n string commitments of length $n-1$, we can build $n(n-1)/2$ bit commitments from which one can be opened. When choosing $s = n/2$, we obtain an exponential number of committed bits from n strings ($N > \frac{2^{n/2}}{\sqrt{n}}$). Note that the protocol is optimal in the length of the strings up to a factor s; otherwise it would be possible to implement a string commitment of length bigger than $n \cdot \ell$ from n instances of string commitment of length ℓ, which is not possible.

Corollary 17 *Let q be a prime power, $d < q$ and $N := q^{d+1}$. There exists a protocol that UC-implements $\mathcal{F}_{RCOM}^{N,r}$ from $(rd+1)q$ instances of $\mathcal{F}_{SCOM}^{N/q}$.*

To obtain an exponential number of bit commitments from n string commitments, we can use Corollary 1 in [DBV03] which gives an explicit construction of a $t-\mathrm{CFF}(\mathcal{X}, \mathcal{B})$ where $|\mathcal{X}| < 24t^2 \log(|\mathcal{B}| + 2)$. Hence,

Corollary 18 *There exists a protocol that realizes $\mathcal{F}_{RCOM}^{N,r}$ from $24r^2 \log(N+2)$ instances of $\mathcal{F}_{SCOM}^{N}$.*

This is close to the optimal efficiency we can expect from Protocol 2, as it has been shown in Theorem 1.1 in [SWZ00] that $t-\mathrm{CFF}(\mathcal{X}, \mathcal{B})$ must have

$$|\mathcal{X}| \geq c \cdot \frac{t^2}{\log t} \log |\mathcal{B}| \ ,$$

for a constant c.

Our protocols can be generalized in a simple way as follows: let $\mathcal{F}_{\mathtt{RCOM}}^{N,r,c}$ be the same functionality as $\mathcal{F}_{\mathtt{RCOM}}^{N,r}$ except that every bit is replaced by a block of size c. The sender can open up to r blocks, or all N blocks at the same time. It is not difficult to see that if Protocol 2 implements $\mathcal{F}_{\mathtt{RCOM}}^{N,r}$ from n instances of $\mathcal{F}_{\mathtt{SCOM}}^{\ell}$, then it can be transformed into a protocol that implements $\mathcal{F}_{\mathtt{RCOM}}^{N,r,c}$ from n instances of $\mathcal{F}_{\mathtt{SCOM}}^{\ell c}$.

4.4.2 Commitments from Noisy Channels at a Constant Rate

Choosing $d = 1$ in Corollary 17, we get $N = q^2$ and $n = (r+1)q$. Thus, there exists a protocol that uses $(r+1)q$ string commitments of length q and implements q^2 bit commitments from which r can be opened. In combination with the string commitment protocol presented in [WNI03], we get the following corollary.

Corollary 19 *For any constant r, there exists a protocol that implements $\mathcal{F}_{\texttt{RCOM}}^{n,r}$ using only $O(n)$ noisy channels.*

This is optimal up to a constant factor.

4.5 Conclusions

In this work we have shown a strong lower bound for reductions of multiple bit commitments to other information theoretic primitives, such as oblivious transfer or noisy channels. Our bound shows that every single bit commitment needs at least $\Omega(k)$ instances of the underlying primitive. This makes bit commitments often much more costly to implement than oblivious transfer, for example.

We have presented protocols that implement bit commitments more efficiently, when the number of bits that can be opened is restricted. Our protocols implement commitments with restricted openings from string commitments. We think that for some resources more efficient protocols might be possible by implementing them directly, instead of using string commitments as a building block.

CHAPTER 5

TRADING ROBUSTNESS FOR CORRECTNESS AND PRIVACY IN CERTAIN MULTIPARTY COMPUTATIONS, BEYOND AN HONEST MAJORITY

Authors: Anne Broadbent, Stacey Jeffery, Samuel Ranellucci, Alain Tapp

5.1 Abstract

We improve on the classical results in information-theoretically secure multiparty computation among a set of n participants, by considering the special case of the computation of the *addition* function over binary inputs in the secure channels model with a *simultaneous* broadcast channel. This simple function is a useful building block for other applications. The classical results in multiparty computation show that in this model, every function can be computed with information-theoretic security if and only if less than $n/2$ participants are corrupt. In this chapter we show that, under certain conditions, this bound can be overcome.

More precisely, let $t^{(p)}, t^{(r)}$ and $t^{(c)}$ be the *privacy*, *robustness* and *correctness* thresholds; that is, the minimum number of participants that must be actively corrupted in order for privacy, robustness or correctness, respectively, to be compromised. We show a series of novel tradeoffs applicable to the multiparty computation of $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n$ for $x_i \in \{0, 1\}$, culminating in the most general tradeoff: $t^{(p)} + t^{(r)} = n + 1$ and $t^{(c)} + t^{(r)} = n + 1$. These tradeoffs are applicable as long as $t^{(r)} < n/2$, which implies that, at the cost of reducing robustness, privacy and correctness are achievable despite a dishonest majority (as an example, setting the robustness threshold to $n/3$ yields privacy and correctness thresholds of $2n/3 + 1$).

We give applications to information-theoretically secure voting and anonymous message transmission, yielding protocols with the same tradeoffs.

Keywords: multiparty computation, secret sharing, information-theoretic security, simultaneous broadcast, addition, voting, anonymous communication.

5.2 Introduction

Secure multiparty computation [Yao82] enables a group of n participants to collaborate in order to compute a global function on their private inputs. Assuming that private random keys are shared between each pair of participants, every function can be securely computed if and only if less than $\frac{n}{3}$ participants are corrupt. This fundamental result is due to Chaum, Crépeau and Damgård [CCD88] and to Ben-Or, Goldwasser and Wigderson [BOGW88]. When a broadcast channel is available, the results of Rabin and Ben-Or [RBO89] tell us that this proportion can be improved to $\frac{n}{2}$.

In [BT07] and [BT08], Broadbent and Tapp presented multiparty protocols for voting and anonymous message transmission that are information-theoretically secure even in the presence of a dishonest majority. Along with the use of authenticated private communication, the protocol uses a simultaneous broadcast channel. In this paper we present a new approach in the same model that achieves better functionality.

In this chapter we show how to achieve tradeoffs between the privacy, correctness, and robustness thresholds for certain multiparty functions. In most multiparty computation results to date, the approach has been to define a model in which bounds on the numbers of corrupt players are known, and to define protocols that work in that model. In another approach, sometimes called hybrid security or multiple threshold security, no assumptions are made on the number or type of adversaries, but rather, various thresholds are given for different adversarial situations. This model reflects reality — it is never known in practice how many participants are honest. It is simply hoped that enough are honest for the security properties of the protocol to hold, some of which may be more important than others.

One way to accomplish this is to differentiate between various types of corrupt participants as is done by Fitzi, Hirt, Holenstein and Wullschleger in [FHHW03] and Fitzi, Hirt and Maurer in [FHM98]. In the latter model, there are t_a actively corrupt players, whose behaviour is entirely controlled by the adversary, t_p passively corrupt players, whose entire information is known to the adversary, and t_f fail-corrupt players, who can be made to cease all participation in the protocol by the adversary, but are otherwise honest. Their results state that multiparty computation with zero failure probability can be done if and only if $3t_a + 2t_p + t_f < n$ (whether or not a broadcast channel is available), and multiparty computation with exponentially small failure probability is achievable given a broadcast if and

only if $2t_a + 2t_p + t_f < n$; if no broadcast is available, the additional condition $3t_a + t_f < n$ is necessary and sufficient.

In a recent result by Lucas, Raub and Maurer [LRM10], tradeoffs were given between information-theoretically secure robustness and computationally secure correctness and privacy for general multiparty computation, achieving the bounds established in [IKLP06b]. We achieve a similar tradeoff, with information-theoretic correctness and privacy, that can be applied to a limited number of multiparty computation problems. The result of [LRM10] is achieved using a technique called *virtual players*, where a set of participants simulate a new participant. We use another kind of virtual player called a *ghost* to achieve our tradeoffs.

5.2.1 Contributions

We define the function $\mathsf{SUM} : \{0,1\}^n \to \{0,\ldots,n\}$ by $\mathsf{SUM}(x_1,\ldots,x_n) = x_1 + \cdots + x_n$, the *integer* sum of n bits. If the input is regarded as an n-bit string, this function is the Hamming weight. We present three protocols for multiparty computation of SUM. Our protocols have the property that the outcome is always correct if all participants are honest.

The protocols can be trivially generalized to allow each participant an input in $\{0,\ldots,k\}$ for arbitrary k by having each participant simulate k participants. Each of our protocols for multiparty sum yields protocols for voting and anonymous message transmission with the same security properties.

The first protocol achieves privacy and correctness in the presence of a dishonest majority of up to $n-1$ actively corrupt participants, but has low robustness: a single participant can make the protocol abort. The corresponding voting protocol improves over the one from [BT07] and [BT08] by having an exact tally.

The second protocol trades privacy for correctness and robustness. For any $t \in [0, \frac{n}{2})$, the protocol is private whenever there are less than $n-t$ corrupt participants, and also correct and robust whenever there are less than $t+1$ corrupt participants. This tradeoff is also applicable to the computation of *any* multiparty linear function. The third protocol achieves a similar but slightly improved tradeoff: privacy and correctness for robustness. For any $t \in [0, \frac{n}{2})$, the protocol is private *and correct* whenever there are less than $n-t$ corrupt participants, and also robust whenever there are less than $t+1$ corrupt participants. The corresponding voting and anonymous message transmission protocols are an improvement over [BT07] and [BT08] in that robustness can be improved at the cost of a slight decrease

in privacy and correctness. It may certainly be the case, particularly in an application such as voting, that privacy and correctness are much more important than robustness, however a robustness threshold greater than 1 may be desirable.

We begin by describing our model in **Section 5.3**. We then give some preliminaries, followed by our three protocols for multiparty sum in **Sections 5.5**, **5.6**, and **5.7**. Finally we show in **Section 5.8** how those protocols can be generalized to larger inputs and how they can be turned into protocols for voting and anonymous message transmission.

5.3 Model and Definitions

In this section, we describe our model and give basic security definitions. For all our security definitions we assume an active adversary that completely controls a certain number of participants. We assume that all pairs of participants are connected by a private authenticated channel, which is equivalent to the assumption that they share a polynomial-sized private random key. We also assume that the participants have access to a simultaneous broadcast channel.

Definition 20 An n participant simultaneous broadcast channel is a collection of n broadcast channels, one for each participant, such that each participant chooses his input to the broadcast before receiving the value of any other participant's broadcast.

It is not uncommon in multiparty computation to allow additional resources, even if those resources cannot be implemented with the threshold on the honest participants (the results of [RBO89], which combine a broadcast channel with $\frac{n}{2}$ honest participants being one example). Our work suggests that a simultaneous broadcast channel is an interesting primitive to study in this context. Sealed bid envelopes that are opened publicly are an example of a practical implementation of a simultaneous broadcast channel. Our protocols then provide *everlasting* security: as long as the computational assumptions are not broken *during* the execution of the protocol (more precisely, during the execution of the simultaneous broadcast), the security of the protocols is perfect. Note that breaking the computational assumption is not sufficient on its own to compromise privacy of the protocols.

Note that under the assumption that trapdoor one-way permutations exist, [FGH$^+$02] gives a protocol for secure multiparty computation in our model; the advantage of our scheme is that we only require simultaneous broadcast.

Throughout this paper, we assume that n is even. We will use calligraphic script letters to denote the players involved in various schemes, such as $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$. Arrays will be denoted using standard vector notation $\vec{x}$ and array elements will be denoted with superscripts: $\vec{x} = (\vec{x}^{(1)}, \ldots, \vec{x}^{(\ell)})$. The notation $[y_1, \ldots, y_n] = \mathsf{SCHEME\text{-}Stage}[\mathcal{P}_1(x_1), \ldots, \mathcal{P}_n(x_n)]$ means that some stage, Stage, of some scheme, SCHEME, is being carried out, with players $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Player i uses input x_i, and receives output y_i.

We will now present the main security definitions.

Definition 21 A multiparty protocol for computing f is *private* if a group of corrupt participants, C, can learn no more about $x_1, \ldots, x_n$ than they would learn from $f(x_1, \ldots, x_n)$ for some choice of $\{x_i : i \in C\}$.

The following two security properties, correctness and robustness, are generally both included in correctness. However, we view them as separate properties in light of the fact that obtaining an incorrect answer is often more problematic than aborting.

Definition 22 A multiparty protocol for computing f is *correct* if (except with exponentially small probability), whenever the protocol does not abort, the output is consistent with the inputs of the honest participants and some fixed inputs for the dishonest participants, known to them before they learn the outcome of the protocol.

Definition 23 A multiparty protocol for computing f is *robust* if it is correct and does not abort except with exponentially small probability.

In the case of a protocol aborting, we can view the output as NULL. If a subprotocol aborts, then by default the calling protocol aborts unless otherwise specified. Note that aborting conditionally on some honest player's input is considered to be breaking privacy.

Definition 24 We denote by $t^{(c)}$ the correctness threshold, or the minimum number of corrupt participants that can compromise correctness. Similarly, $t^{(p)}$ denotes the privacy threshold, and $t^{(r)}$ the robustness threshold.

Unlike in [FHM98], we only consider an active adversary; one with complete control over the actions of each player it corrupts. We do not consider an adversary who actively corrupts some amount of players and then passively corrupts some additional players. Additionally, though our notation is similar to [FHM98], the meaning is quite different. In [FHM98],

the meaning of t_p is the number of participants that are passively corrupted, whereas in our model, $t^{(p)}$ is the minimum number of participants that must be actively corrupted for privacy to be lost.

We do not place any restrictions on the dishonest participants, though we assume that all corrupt participants are part of a single collusion, called the adversary.

5.4 Preliminaries

5.4.1 Sharing a Secret

All of our protocols are based on secret sharing [Sha79], and derive their security properties in part from the secret sharing scheme used. A secret sharing scheme is a multiparty computational primitive, whereby a secret can be distributed over a group of participants such that an authorized group of participants can reconstruct the secret (correctness), and any unauthorized group of participants can learn nothing about the secret (privacy).

Note that the notions of privacy and correctness for secret sharing are slightly different than those for general multiparty computation, however we still use $t^{(p)}$, $t^{(c)}$, and later $t^{(r)}$, to denote the thresholds for privacy, correctness and robustness, respectively.

What defines an authorized group varies between secret sharing schemes. For instance, in some secret sharing schemes, an authorized subset of participants is defined to be any set of more than t participants, for some $t \leq n$. Such a secret sharing scheme is called a t-out-of-n threshold secret sharing scheme.

The first protocol, $\mathsf{SS}_{p,n}$ (see **Scheme 1**), uses a very basic n-out-of-n threshold secret sharing scheme. A secret sharing scheme has two phases, *distribute* and *reconstruct*. The distribution phase is a protocol for constructing shares of the secret and distributing them to the receivers. The reconstruction phase is a protocol by which an authorized set of receivers can learn the secret.

Privacy in $\mathsf{SS}_{p,n}$ follows from the fact that, given any group of $n-1$ participants $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \setminus \mathcal{P}_j$, there are p equiprobable possibilities for $\mathcal{P}_j$'s share, each corresponding to a distinct possibility for the secret $m \equiv m_j + \sum_{i \neq j} m_i \pmod{p}$. Therefore, given $n-1$ shares, the secret is still completely unknown.

Definition 25 We say that a secret sharing scheme SCHEME is *linear* if, for a publicly

Scheme 1 $\mathsf{SS}_{p,n}$

Players: a sender $\mathcal{S}$
n receivers $\mathcal{R}_1, \ldots, \mathcal{R}_n$

Distribute: $[\emptyset, m_1, \ldots, m_n] = \mathsf{SS}_{p,n}\mathsf{Dist}[\mathcal{S}(m), \mathcal{R}_1, \ldots, \mathcal{R}_n]$
Input: m, the secret, input by sender $\mathcal{S}$
Output: m_i, the ith share, output to $\mathcal{R}_i$ for each i, such that $\sum_{i=1}^{n} m_i = m \bmod p$

1. $\mathcal{S}$ chooses $m_i \in_R \mathbb{Z}_p$ for $i = 1, \ldots, n-1$, and sets $m_n = m - \sum_{i=1}^{n-1} m_i \bmod p$
2. $\mathcal{S}$ sends m_i to $\mathcal{R}_i$

Reconstruct: $[m, \ldots, m] = \mathsf{SS}_{p,n}\mathsf{Rec}[\mathcal{R}_1(m_1), \ldots, \mathcal{R}_n(m_n)]$
Input: m_i input by $\mathcal{R}_i$ for $i = 1, \ldots, n$
Output: m output to $\mathcal{R}_i$ for $i = 1, \ldots, n$

1. Each $\mathcal{R}_i$ for $i = 1, \ldots, n$, inputs m_i into the simultaneous broadcast channel.
2. Each $\mathcal{R}_i$ constructs $m = \sum_{j=1}^{n} m_j \bmod p$

---

known integer a and any two secrets m and m' shared among $\mathcal{P}_1, \ldots, \mathcal{P}_n$, with shares $m_1, \ldots, m_n$ and $m'_1, \ldots, m'_n$ respectively, if $\{\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_t}\}$ is an authorized subset of receivers, then $\mathsf{SCHEME\text{-}Rec}[\mathcal{P}_{i_1}(am_{i_1} + m'_{i_1}), \ldots, \mathcal{P}_{i_t}(am_{i_t} + m'_{i_t})]$ outputs $am + m'$ to each $\mathcal{P}_{i_j}$.

It is easy to see that $\mathsf{SS}_{p,n}$ is a linear secret sharing scheme.

Definition 26 A *linear distributed secret*, $\mathsf{LDS}[\mathcal{P}_1, \ldots, \mathcal{P}_n](m)$ is a list of shares (m_i), each in possession of player $\mathcal{P}_i$, such that $[m_1, \ldots, m_n] = \mathsf{SCHEME\text{-}Dist}[m]$ for some linear secret sharing scheme SCHEME.

5.4.2 Sub-Protocols Used

Given a set of two or more linear distributed secrets $\mathsf{LDS}[\mathcal{R}_1, \ldots, \mathcal{R}_n](m^{(j)})$, relative to any linear secret sharing scheme, the participants can always create a new linear distributed secret by the following: each $\mathcal{R}_i$ adds all his shares to get a share of $\sum_j m^{(j)}$. This simple procedure does not involve any interaction between participants, but allows them to generate a new shared secret: the sum of two or more previously shared secrets.

Procedure 1 shows how to generate randomness in a group of participants $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ with a simultaneous broadcast. As long as one participant is honest, the output is an unbiased integer between 0 and $p - 1$.

Procedure 1 $[a, \ldots, a] = \mathsf{RANDOM}_{n,p}[\mathcal{P}_1, \ldots, \mathcal{P}_n]$

Players: n participants, $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Output: An unbiased $a \in \mathbb{Z}_p$ is output to all players

1. Each participant $\mathcal{P}_i$ inputs $a_i \in_R \{0, ..., p-1\}$ into the simultaneous broadcast channel.
2. Each participant $\mathcal{P}_i$ sets $a = \sum_{i=1}^{n} a_i \pmod{p}$.

In order to bias the outcome, a corrupt participant's input would have to depend on the inputs of all other players. The simultaneous broadcast channel makes this impossible.

The following procedure can be used to check the equality of a set of linear distributed secrets, without revealing the values of the individual secrets. If there are two distinct secrets in the set, then the procedure outputs unequal except with exponentially small probability in the security parameter s. This procedure is used in our second and third protocols.

Procedure 2 $\mathsf{EQUALITY}_s$

Players: n participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$
Input: $\{X^j\}_{j=1}^{2s}$ a set of $2s$ LDSs, $X^j = \mathsf{LDS}(x^j)$
Output: equal or unequal

Repeat the following s times in parallel:

1. The participants use RANDOM to choose a random partition $\{P, Q\}$ of $[1..2s]$ with $|P| = |Q| = s$.
2. The participants locally compute $Y = \sum_{j \in P} X^j - \sum_{j \in Q} X^j$.
3. The participants reconstruct the secret Y. If $Y \neq 0$ they output unequal.

If $Y = 0$ in every round, output equal.

5.4.3 Properties of GHOST-SUM

Theorem 27 EQUALITY *detects inequality* $\pmod{p}$ *in* $\{X^i\}$*, except with exponentially small probability.*

Proof.

Suppose the input $\{X^j\}$ is unequal (the values distributed are not all the same). Let P and Q be any partition of $\{X^j\}$ such that $|P| = |Q|$ and $\sum_{i \in P} X^i = \sum_{i \in Q} X^i$. Note that by swapping any two non-equal elements in Q and P respectively, we make the two sums unequal. This observation is not entirely obvious, since we're working $\pmod{p}$. Suppose we swap $a \in P$ and $b \in Q$ where $a > b$. This will result in $\sum_{i \in P} X^i$ decreasing by $a - b$ and

$\sum_{i \in Q} X^i$ increasing by $a - b$. Since these two sums were equal before the swap, we now have a difference of $2(a-b)$. If $2(a-b) \equiv 0 \pmod{p}$, since p is odd, then we must have $(a-b) \equiv 0 \pmod{p}$, which is a contradiction since a and b are assumed non-equal. So as long as we swap non-equal elements from the partitions, their sums will no longer be equal. From this observation, we will show that there are at least as many partitions with $\sum_{i \in P} X^i \neq \sum_{i \in Q} X^i$ as there are with $\sum_{i \in P} X^i = \sum_{i \in Q} X^i$.

Consider the operation of swapping the first two unequal elements in the sorted sets P and Q. Clearly this operation maps equal sum partitions to unequal sum partitions. In addition, let us specify that the partition where P and Q are identical be mapped to the partition obtained by sorting the set $\{X^i\}$ and setting P equal to the first half; the result will be unequal since $\{X^i\}$ is unequal. We now have a one-to-one mapping from equal sum partitions to unequal partitions, so no more than half of the possible partitions can have the property $\sum_{i \in P} X^i = \sum_{i \in Q} X^i$.

Thus the probability of choosing a partition with this property when two or more elements are unequal is less than $\frac{1}{2}$. With s repetitions, the probability of an unequal set passing $\mathsf{EQUALITY}$ is less than $\frac{1}{2^s}$. □

Theorem 28 $t^{(p)}(\mathsf{GHOST\text{-}SUM}_{n,g}) = \frac{n+g}{2}$.

Proof. Again, the privacy threshold follows directly from that of the employed secret sharing scheme. □

Theorem 29 $t^{(c)}(\mathsf{GHOST\text{-}SUM}_{n,g}) = t^{(r)}(\mathsf{GHOST\text{-}SUM}_{n,g}) = \frac{n-g}{2}$.

Proof. The protocol aborts if and only if:

1. the honest participants fail to reconstruct the total (secret sharing scheme aborts), or
2. the reconstructed total's bins sum to a value other than n.

The first case is not possible, by **Theorem 35**.

Note that the value of the secret cannot be changed after the distribution phase because the total number of non-honest parties is less than $\frac{n-g}{2} + g = \frac{n+g}{2}$, which is $t^{(c)}(\mathsf{ICVSS}_{n+g})$.

We therefore need only show that if an input is ill-formed it will be found in the verification steps and discarded and if an input differs across the s parallel computation rounds it will be found in the second verification step.

If a participant wants to share an ill-formed BIN-ARRAY in a set, he must create no more than s invalid BIN-ARRAYs in that set, or he is guaranteed to have at least one invalid BIN-ARRAY opened in the first verification phase. If he has $1 \leq x \leq s$ invalid BIN-ARRAYs, then the probability that no invalid BIN-ARRAY is opened is:

$$\frac{\binom{2s-x}{s}}{\binom{2s}{s}} \leq \frac{1}{2}$$

If the participant has at least one invalid BIN-ARRAY per set, the probability that an invalid BIN-ARRAY is not opened is $\leq \frac{1}{2^s}$. If a participant has invalid BIN-ARRAYs in some sets and not others, then his inputs are not equal across all rounds.

If the inputs are not equal across all rounds, then except with exponentially small probability, the input is discarded and the player disqualified in the second verification step. □

5.5 Multiparty Sum with Bins

The first tool we will apply is the use of a concept we call *bins*. The intuitive description that follows makes clear the reason for this name.

The following physical analogy applies to all three of our protocols. The protocols are modelled after the concrete setup of an array of $2n$ bins. A participant may place a ball in any bin, but may not remove a ball from a bin or observe the contents of a bin. If $x_i = 1$, participant $\mathcal{P}_i$ chooses a random bin from $j = 1, \ldots, n$, called the *count bins*, and places his ball in the jth bin. Otherwise, if $x_i = 0$, $\mathcal{P}_i$ chooses a random bin from $j = n+1, \ldots, 2n$, called the *no-count bins*, and places his ball in the jth bin. When all balls have been placed, the totals for each bin are revealed and the sum over all balls in the count bins (the first n bins) is the output $y = \mathsf{SUM}(x_1, \ldots, x_n)$. So far the need for $2n$ bins instead of 2 bins is not clear, but we will soon explain this necessity.

For our protocols, we model each bin as an integer $\pmod p$, with an input x_i encoded as a string of $2n$ integers, one integer for each bin ($p \geq 2n+1$). The ith integer of an encoded input represents the contents of the ith bin. In this case, a well-constructed input encoding has exactly one bin with value 1 and all other bins with value 0. In our protocols, each participant splits his input into shares, each share consisting of $2n$ integers $\pmod p$, with the property that each bin of the input array can be reconstructed from the bins of the shares

using some secret sharing scheme. For instance, if we use SS, the bin-wise sum $\pmod p$ of the shares is equal to the encoded input. Given a set of bin array inputs shared among the n participants, it is easy to compute the tally without revealing any information about the inputs, by simply adding the shares and reconstructing the total. We require only that the secret sharing scheme used be linear.

Without looking at individual bin arrays, we must ensure that all bin arrays are well-constructed. If a participant attempts to contribute more than 1 ball to the total, or negate part of the total by a constant c by putting $p - c$ balls in the n count bins, then the sum over all $2n$ bins in the tally will not be equal to n. Thus, a cheating strategy would be to include $c + 1$ balls in the n count (respectively no-count) bins, and $-c = p - c$ balls in the n no-count (respectively count) bins. However, having $2n$ bins makes it likely that many bins will be empty and a negative number of balls, $p - c$, in an empty bin would be detected, since $p - c > n$. This justifies the need for $2n$ bins, as well as $p = 2n + 1$. A negative number of balls is detected with constant probability and repetition yields exponential security.

5.5.1 Protocol

Here we present the first protocol for SUM, which makes use of the bins idea. It may be useful to consider the simple secret sharing scheme, SS, but note that any linear secret sharing scheme would work.

The following procedure encodes a bit as described above.

Procedure 3 BIN-ARRAY$_n$

Input: $x \in \{0, 1\}$

Output: $\vec{x} = (\vec{x}^{(i)})_{i=1}^{2n}$

1. if $x = 1$ choose $j \in_R \{1, \ldots, n\}$ otherwise if $x = 0$ chose $j \in_R \{n + 1, \ldots, 2n\}$
2. for $i = 1, \ldots, 2n$: if $i = j$, $\vec{x}^{(i)} = 1$, else $\vec{x}^{(i)} = 0$

We call an array encoding an integer in this way a BIN-ARRAY. We say that a BIN-ARRAY is *well-formed* if each bin is an integer in $\{0, 1\}$ and the sum over all bins is 1. A BIN-ARRAY that is not well-formed is called *ill-formed*.

We will need to distribute BIN-ARRAYs among the n participants. We can easily distribute shares of an array by simply creating shares of each entry. We define the following addition

on shares of an array, which follows directly from the addition on the individual shares. Let $\vec{a}_i = (\vec{a}_i^{(j)})_{j=1}^{\ell}$ and $\vec{b}_i = (\vec{b}_i^{(j)})_{j=1}^{\ell}$ be two array-shares. Then $\vec{a}_i + \vec{b}_i = (\vec{a}_i^{(j)} + \vec{b}_i^{(j)})_{j=1}^{\ell}$.

We now give our first protocol for SUM.

Protocol 1 $\mathsf{BIN\text{-}SUM}_{n,s}$

Players: n participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Input: $\mathcal{P}_i$ inputs $x_i \in \{0, 1\}$, for $i = 1, \ldots, n$

Output: Each participant gets output $y = x_1 + \cdots + x_n$

Repeat in parallel s times:

1. Each $\mathcal{P}_i$ creates $\vec{x}_i = \mathsf{BIN\text{-}ARRAY}_n(x_i)$ and distributes $X_i = \mathsf{LDS}[\mathcal{P}_1, \ldots, \mathcal{P}_n](\vec{x}_i)$ using some linear secret sharing scheme.
2. Participants create the sum, $Y = \sum_{j=1}^{n} X_j$.
3. Participants input their shares of Y into the simultaneous broadcast channel and reconstructs the value of Y, $\vec{y}$.
4. If the sum over all bins in $\vec{y}$ does not equal n, abort.
5. Each participant computes $y = \sum_{j=1}^{n} \vec{y}^{(j)}$, the total over all count bins of Y.

If the outcome y is not the same in every round, abort.

Since the only way to break privacy is to break the secret sharing scheme, as long as we use a secret sharing scheme with $t^{(p)} = n$, such as SS, we get $t^{(p)}(\mathsf{BIN\text{-}SUM}) = n$ (**Theorem 30**).

The only way for a adversary to make the output inconsistent with the honest inputs is to put more than 1 ball in either the count or the no-count bins. If they use more than 1 ball in total, it will be detected when the total number of balls in y is more than n, and the protocol will abort. To avoid this, the adversary must put -1 in some bin, and hope that it is non-empty. However, any bin is empty with constant probability, and so repetition yields exponential security. Therefore, no dishonest coalition of any size can make an output that is inconsistent with honest inputs, that is, $t^{(c)} = n$ (**Theorem 31**).

The major downfall of this protocol is that any participant can make the protocol abort. In other words, $t^{(r)} = 1$ (**Theorem 32**). In many situations, such as voting (see **Section 5.8.1**), it may be desirable to have $t^{(p)}$ and $t^{(c)}$ much higher than $t^{(r)}$, however, it is often desirable to have $t^{(r)} > 1$. In our next protocol, we allow $t^{(r)}$ to be as high as $\frac{n}{2}$ at the expense

of some privacy and correctness.

5.5.2 Properties of BIN-SUM

Theorem 30 $t^{(p)}(\mathsf{BIN\text{-}SUM}) = n$.

Proof. The privacy threshold of BIN-SUM follows directly from that of the employed secret sharing scheme. If we use a threshold scheme with threshold n, such as SS, then we get $t^{(p)} = n$ in BIN-SUM. □

Theorem 31 $t^{(c)}(\mathsf{BIN\text{-}SUM}) = n$.

Proof. Suppose a coalition of $c < n$ dishonest voters wishes to cause the sum to be incorrect. If they deposit more than c balls between them, the final tally over all bins will be greater than n and the protocol will abort. Thus, the only way for them to cause the final sum to be inconsistent with the honest inputs is for at least one dishonest participant to put a negative number of balls in at least one bin, say bin b. If no other participant deposits a ball in bin b, then the bin total will be $p - 1 > n$, so the protocol will abort. For a participant to succeed in depositing a negative ball in a count bin (respectively no-count bin), he must put his negative ball in a no-count bin (respectively count bin) with at least one ball in it. Even in the worst case where all $n - 1$ other balls are deposited in the no-count bins, the probability that bin b is empty is $(\frac{n-1}{n})^{n-1}$, which is greater than $\frac{1}{3}$ for all n. By repeating the protocol s times, the probability that a participant successfully deposits a negative ball without the protocol aborting is less than $(\frac{2}{3})^s$. □

Theorem 32 $t^{(r)}(\mathsf{BIN\text{-}SUM}) = 1$.

Proof. A single participant need only encode a number that is strictly greater than 1 to cause the total number of balls in all bins to be strictly greater than n, making the protocol abort. □

5.6 Multiparty Sum with Bins and Ghosts

The second tool we make use of is the concept of *ghost players*. Given a verifiable secret sharing scheme with privacy, correctness and robustness thresholds $t^{(p)} = t^{(c)} = t^{(r)} = \frac{n}{2}$,

we modify the protocol as follows. During the distribution phase, the sender creates $n + g$ shares, distributes n of them, and discards g of them, for some $g \in [0, n)$. This lowers the proportion of actively corrupt participants by adding participants who cannot be actively corrupted. Now $\frac{n+g}{2}$ corrupt players are required to break privacy, and ghost players cannot contribute to this corrupt coalition. Thus we get privacy threshold $t^{(p)} = \frac{n+g}{2}$. In order to prevent the correct reconstruction of a secret, we need $\frac{n+g}{2}$ corrupt players. We can't assume the ghosts do not contribute to such a collusion, so among the real players, we have correctness and robustness thresholds of $t^{(c)} = t^{(r)} = \frac{n+g}{2} - g = \frac{n-g}{2}$. We thus obtain a tradeoff between correctness and robustness, $t = \frac{n-g}{2}$, and privacy, $n - t$, where $t \in [0, \frac{n}{2})$. Again, our protocol implements the multiparty computation of SUM, but we can also use the linearity of the underlying secret sharing scheme to show that this tradeoff can be applied to any *linear* function as well. (Note that our function, SUM is not actually linear, because of the restriction that the input be in $\{0, 1\}$). This is accomplished, without the use of bins. Each participant simply distributes his input x_i to some linear function f, using the verifiable secret sharing scheme we now present, GVSS. Each participant computes f on his shares of the inputs and outputs the resulting share of $f(x_1, \ldots, x_n)$.

We can view the ghosts as a kind of virtual player. Since these ghost players do nothing, they are all fail-corrupt. However, they are honest in every other respect.

We do require that the secret sharing scheme to which we apply the ghost modification be verifiable to gain the desired accuracy and robustness thresholds. We now detail this concept.

5.6.1 Verifiable Secret Sharing

Verifiable secret sharing is an extension of secret sharing that allows reconstruction of a secret even in the presence of faulty or missing shares. More formally, a verifiable secret sharing scheme is a secret sharing scheme with the following properties:

P1 If the sender is honest, then before the reconstruction phase has been initiated, the adversary has no information about the secret.

C1 The probability that the distribution phase completes successfully and there exist distinct m and m' such that both m and m' have non-negligible probability of being the outcome of the reconstruction phase is exponentially small. If the sender is honest, the unique m that can be the outcome of reconstruction with non-negligible probability

is the message input by the sender during the distribution phase.

R1 The probability that the distribution succeeds and the reconstruction phase outputs NULL is exponentially small (even if the sender is corrupted). If the sender is honest, the distribution phase succeeds except with exponentially small probability.

In any verifiable secret sharing scheme, each property will be subject to some threshold of corrupt participants. Property **R1** and **C1** are often combined, but we will find it convenient to discuss them separately.

It is not difficult to see that these properties are necessary to obtain our tradeoff, for otherwise we could not reconstruct a non-NULL secret from the uncorrupted shares.

There exist linear verifiable secret sharing schemes with $t^{(c)} = t^{(r)} = t^{(p)} = \frac{n}{2}$ [CDM00]. We can use such a scheme, in conjunction with the usage of ghost players, to get an improved protocol for SUM. The protocol, which we call GHOST-SUM, is similar to BIN-SUM, but we require the secret sharing scheme used to be a verifiable secret sharing scheme, and we modify the distribution phase slightly. In addition, we add some verification steps to ensure that the inputs are well-formed in order to increase robustness.

Let VSS_n be a black box $\frac{n}{2}$-out-of-n-threshold verifiable secret sharing scheme (satisfying properties **P1**, **C1**, and **R1**) with distribution phase $[\emptyset, m_1, \ldots, m_n] = \mathsf{VSS}_n\mathsf{Dist}[\mathcal{S}(m), \mathcal{R}_1, \ldots, \mathcal{R}_n]$. We define the distribution phase of $\mathsf{GVSS}_{n,g}$ as:

$$\mathsf{GVSS}_{n,g}\mathsf{Dist}[\mathcal{S}(m), \mathcal{R}_1, \ldots, \mathcal{R}_n] = \mathsf{VSS}_{n+g}\mathsf{Dist}[\mathcal{S}(m), \mathcal{R}_1, \ldots, \mathcal{R}_n, \mathcal{S}, \ldots, \mathcal{S}]$$

In words, the distribution phase of $\mathsf{GVSS}_{n,g}$ is just the distribution phase of VSS_{n+g}, where $\mathcal{S}$ does not distribute the last g shares to other players. These shares are considered to be discarded.

5.6.2 Properties of GVSS

Theorem 33 *Suppose an honest sender has distributed a secret m using any verifiable secret sharing scheme with privacy threshold $t^{(p)}$. Then a coalition of $t^{(p)} - 1$ actively corrupt parties and any number of fail-corrupt participants can't gain any information about m.*

Proof. Suppose the actively corrupt participants could learn some information about m. This information must be a function of the private communication transcript for each corrupt

player, as well as the transcript of all public broadcasts. A fail-corrupt participant adds no information to these, since he does not communicate at all. Therefore, if the actively corrupt participants could gain information about m in the presence of fail-corrupt players, they could do so without the fail-corrupt players, contradicting the threshold $t^{(p)}$. □

Theorem 34 *Suppose an honest sender has distributed a secret m using $\mathsf{GVSS}_{n,g}$. Then a coalition of less than $\frac{n+g}{2}$ dishonest participants can't learn any information about m except with exponentially small probability.*

Proof. There are at most $\frac{n+g}{2} - 1$ actively corrupt participants, out of $n + g$ participants total. The rest are either fail-corrupt (ghosts) or honest. therefore, by **Theorem 33**, the actively corrupt participants can learn no information about m. □

Theorem 35 *Suppose less than $\frac{n-g}{2}$ participants are corrupt. The probability that the distribution phase of $\mathsf{GVSS}_{n,g}$ outputs shares $m_1, \ldots, m_n$ to receivers $\mathcal{R}_1, \ldots, \mathcal{R}_n$ and there is no fixed m such that the reconstruction phase will output m is exponentially small.*

Proof. Suppose there are $\frac{n-g}{2} - 1$ corrupt participants. There are $n - (\frac{n-g}{2} - 1) = \frac{n+g}{2} + 1$ honest participants, so with probability exponentially close to 1, the distribution phase aborts, or it succeeds and there exists a fixed value m such that $\mathsf{VSS}_{n+g}\mathsf{Rec}$ will output m except with exponentially small probability, by properties **C1** and **R1** of VSS (with threshold $\frac{n}{2}$ for n participants). □

In our second protocol, the use of a verifiable secret sharing scheme means that the participants commit to their inputs in some way. In BIN-SUM with SS, a participant can change his input any time he wants in an arbitrary way by simply changing his own share of his input. In contrast, after the distribution stage of a verifiable secret sharing scheme, the sender is committed to his secret. Therefore, in this second protocol, we introduce some verification steps where the participants check that the committed BIN-ARRAYs are well-formed without learning their values. This allows us to raise the robustness threshold by detecting and eliminating ill-formed BIN-ARRAYs.

One tool we will use in the verification is equality testing. This allows participants to check that in every round the inputs from a particular participant have the same value.

Procedure 4 BIN-ARRAY-EQUALITY$_s$

Players: n participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$
Input: $\{X^j\}_{j=1}^{s^2}$, a set of LDSs, with $X^j = \mathsf{LDS}(\vec{v}_j)$ where $\vec{v}_j = (v_j^{(\ell)})_{\ell=1}^{2n}$ is a BIN-ARRAY
Output: equal or unequal

For $i = 1, \ldots, s-1$:

1. For $j = (i-1)s+1, \ldots, (i+1)s$
   (a) $Y^j = \sum_{\ell=1}^{n} \mathsf{LDS}(v_\ell^{(j)})$
   (b) $N^j = \sum_{\ell=n+1}^{2n} \mathsf{LDS}(v_\ell^{(j)})$
2. Compute EQUALITY$(\{Y^j\}_{j=(i-1)s+1}^{(i+1)s})$ and EQUALITY$(\{N^j\}_{j=(i-1)s+1}^{(i+1)s})$. If either outputs unequal, return unequal.

Return equal.

This procedure takes s sets of s shared BIN-ARRAYs and checks that they encode the same value. It does this by equality testing with two sets at a time, testing the equality of the sum over the count bins, Y^j, as well as the sum over the no-count bins, N^j.

The other verification technique involves opening some BIN-ARRAYs to see that they are well-formed. In order to avoid revealing the value of the opened BIN-ARRAYs, each participant shares his BIN-ARRAYs with the bins permuted. Some of the BIN-ARRAYs are selected for opening, and the sender reveals the permutations on his unopened BIN-ARRAYs so that they can be unpermuted before computation takes place.

Protocol 2 GHOST-SUM$_{n,s,g}$

Players: n participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Input: $\mathcal{P}_i$ inputs $x_i \in \{0, 1\}$, for $i = 1, \ldots, n$

Output: Each participant gets output $y = x_1 + \cdots + x_n$

Preparation Each $\mathcal{P}_i$ creates s *sets* of $2s$ copies of identical BIN-ARRAY$_n(x_i)$s. The BIN-ARRAYs should vary randomly over different sets, but be identical within a set. $\mathcal{P}_i$ then applies a random permutation to each BIN-ARRAY.

Distribution Each BIN-ARRAY is distributed among all n participants using GVSS$_{n,g}$Dist. If any call to GVSS$_{n,g}$Dist aborts, the sender is excluded from future steps of the protocol.

Verification 1 For each set of $2s$ BIN-ARRAYs:

1. Half the BIN-ARRAYs are opened (chosen using RANDOM) and checked for well-formedness.
2. For each unopened BIN-ARRAY in this iteration, $\mathcal{P}_i$ broadcasts the permutations on the bins, which each participant applies to his shares of that BIN-ARRAY.

Verification 2 For each participant $\mathcal{P}_i$, all s^2 of $\mathcal{P}_i$'s unopened BIN-ARRAYs are put into BIN-ARRAY-EQUALITY. If it returns unequal then $\mathcal{P}_i$'s shares are discarded and he is excluded from future steps of the protocol.

Computation For each participant $\mathcal{P}_i$, the participants choose the first unopened BIN-ARRAY from each *set* and use these to compute s parallel totals, as in **Protocol 1**. If each repetition does not give the same answer the protocol aborts.

What we have essentially done here is to throw away the correctness of the first protocol, making it the same property as robustness. We then use the ghost players to establish a tradeoff between privacy and correctness/robustness. The result is a threshold tradeoff $(t^{(p)}, t^{(c)}, t^{(r)}) = (n-t, t+1, t+1)$ for any $t \in [0, \frac{n}{2})$, by setting $g = n - 2t$ (**Theorem 28** and **29**). However, by using a verifiable secret sharing scheme with certain desirable properties, we can keep correctness fairly high, and then use the ghosts to establish a tradeoff between privacy/correctness and robustness. In our third and final protocol, we do just that.

5.7 Multiparty Sum with Bins, Ghosts and Commitments

We now present our third and final protocol for SUM. It has the strongest security properties, with $(t^{(p)}, t^{(c)}, t^{(r)}) = (n-t, n-t, t+1)$.

This new protocol uses the concept of ghosts, just as in the second, but we obtain an improved tradeoff by making use of the particular properties of a specific verifiable secret sharing scheme from [CDD+99a].

5.7.1 Verifiable Secret Sharing with Signatures

We will outline the verifiable secret sharing scheme of [CDD+99a], which we call ICVSS, and show that it has the properties we require to achieve our improved tradeoff.

The scheme makes use of an information-theoretically secure pseudo-signature that has the properties we require. The secret is encoded as $f(0)$ for some degree $\leq \frac{n}{2}$ polynomial f, with shares $f(i)$ for $i \in \{1, \ldots, n\}$. Each player $\mathcal{P}_i$ *commits* to his shares by distributing signed shares of his shares, which we call *subshares*.

The signature scheme involves a signer $\mathcal{S}$, an intermediate receiver $\mathcal{I}$, and several final receivers $\mathcal{R}_1, \ldots, \mathcal{R}_n$. The three stages are as follows:

Distribute $\mathcal{S}$ sends a message m to $\mathcal{I}$, and some auxiliary signature information to $\mathcal{I}$ and each $\mathcal{R}_i$.

Confirm For each i, $\mathcal{I}$ and $\mathcal{R}_i$ carry out computations on their auxiliary information to ensure that if $\mathcal{R}_i$ is honest, he will accept m in the reveal phase.

Reveal $\mathcal{I}$ reveals m to all $\mathcal{R}_i$ and gives each $\mathcal{R}_i$ some auxiliary information. Each $\mathcal{R}_i$ accepts or rejects m. If more than $t^{(r)}$ receivers accept, then m is considered to be accepted.

The signature scheme has the following properties:

SC1 If $\mathcal{S}$, $\mathcal{I}$, and at least $t^{(r)}$ of $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are honest, then each honest $\mathcal{R}_i$ will accept in the reveal phase.

SC2 If $\mathcal{I}$, $\mathcal{R}_i$, and at least $t^{(r)}-1$ other receivers are honest, then after the confirm phase, $\mathcal{I}$ knows a message m such that $\mathcal{R}_i$ will accept m in the reveal phase.

SC3 If $\mathcal{S}$, $\mathcal{R}_i$, and at least $t^{(c)}-1$ other receivers are honest, then $\mathcal{R}_i$ will reject every value $\tilde{m} \neq m$ except with exponentially small probability.

SP1 If $\mathcal{S}$ and $\mathcal{I}$ are honest, then no $\mathcal{R}_i$ can learn any information about m before the reveal phase.

SL1 If c is a publicly known scalar, and σ and σ' represent signatures for messages m and m' respectively, then $c\sigma + \sigma'$ is a signature for $cm + m'$.

A pseudo-signature scheme is a kind of commitment scheme. We say that $\mathcal{S}$ *commits to* m *through* $\mathcal{I}$. Of course, the value of m is only committed to from the perspective of $\mathcal{R}_1, \ldots, \mathcal{R}_n$, who must have received some auxiliary information during the distribute and confirm phases. However, for simplicity, we will gloss over this fact by saying $\mathcal{S}$ commits to m through $\mathcal{I}$ and assume that every participant in the calling protocol is implicitly a receiver. When we say that some $\mathcal{I}$ *opens a commitment* we mean that he carries out the reveal phase.

A pseudo-signature scheme with the above properties, as well as a verifiable secret sharing scheme based on these signatures can be found in [CDD+99a]. Both have thresholds $t^{(r)} = t^{(c)} = t^{(p)} = \frac{n}{2}$. We now present their secret sharing scheme, which will form the basis for our third protocol.

Definition 36 A vector of shares $(m_1, \ldots, m_n)$ is t-consistent if there exists a polynomial f of degree at most t containing every point (i, m_i) for $i = 1, \ldots, n$.

Scheme 2 ICVSS_n

Players: a sender $\mathcal{S}$

n receivers $\mathcal{R}_1, \ldots, \mathcal{R}_n$

Distribute: $[\emptyset, m_1, \ldots, m_n] = \mathsf{ICVSS}_n\mathsf{Dist}[\mathcal{S}(m), \mathcal{R}_1, \ldots, \mathcal{R}_n]$

Input: m, the secret, input by sender $\mathcal{S}$

Output: m_i, the ith share, output to receiver i, $\mathcal{R}_i$

1. $\mathcal{S}$ randomly chooses $f \in \mathbb{Z}_p[x][y]$ of degree at most $\frac{n}{2}$ in each variable such that $f(0,0) = m$. Let $m_{ij} = f(i,j)$. For each $i \in \{1, \ldots, n\}$, $\mathcal{S}$ sends $\mathcal{R}_i$ the vectors $\vec{c}_i = (m_{1i}, \ldots, m_{ni})$ and $\vec{r}_i = (m_{i1}, \ldots, m_{in})$ $\mathcal{S}$ commits to $\vec{c}_i$ and $\vec{r}_i$ through $\mathcal{R}_i$. Set $m_i = (\vec{c}_i, \vec{r}_i)$.
2. Each receiver $\mathcal{R}_i$ checks that the two vectors he received are $\frac{n}{2}$-consistent. If not, $\mathcal{R}_i$ opens the inconsistent vector that $\mathcal{S}$ committed to. If the commitment is accepted, then the protocol aborts.
3. For $i = 1, \ldots, n$ and $j = 1, \ldots, n$ with $j \neq i$, $\mathcal{R}_i$ sends $\mathcal{R}_j$ the share m_{ji}. $\mathcal{R}_i$ commits to m_{ji} through $\mathcal{R}_j$. $\mathcal{R}_i$ ensures that the value he receives from $\mathcal{R}_j$, $\tilde{m}_{ij}$ matches his m_{ij}. If $m_{ij} \neq \tilde{m}_{ij}$ then $\mathcal{R}_i$ opens the commitment to m_{ij} from $\mathcal{S}$ and the commitment to $\tilde{m}_{ij}$ from $\mathcal{R}_j$. Seeing this, $\mathcal{R}_j$ opens the commitment to $\tilde{m}_{ij}$ from $\mathcal{S}$ (or is disqualified). If all commitments are accepted, then the protocol aborts.

Reconstruct: $[m, \ldots, m] = \mathsf{ICVSS}_n\mathsf{Rec}[\mathcal{R}_1(m_1), \ldots, \mathcal{R}_n(m_n)]$

Input: m_i input by $\mathcal{R}_i$ for $i = 1, \ldots, n$

Output: m output to $\mathcal{R}_i$ for $i = 1, \ldots, n$

1. Each $\mathcal{R}_i$ inputs his row $\vec{r}_i = (m_{i1}, \ldots, m_{in})$ into the simultaneous broadcast channel, and for each j, opens the commitment from $\mathcal{R}_j$ to m_{ij}. If any of these commitments are rejected, then $\mathcal{R}_i$ is disqualified. All other players check that this row is $\frac{n}{2}$-consistent.
2. The secret m can be interpolated from the shares of non-disqualified players.

We can characterize the exact property of ICVSS that allows us to achieve our improved tradeoff for SUM. In order to achieve correctness in SUM, we require that the secret sharing scheme have a weaker version of the correctness property, since some of the correctness of

SUM comes from the bin technique and verification steps. The property is as follows:

C′ Suppose the behaviour of corrupt parties results in the reconstruct phase outputting $\tilde{m}$. Before the reconstruction phase has completed they have exactly as much information about $\tilde{m}$ as they have about the secret.

What this property essentially means is that the corrupt participants, though they may be able to change the output of the reconstruction phase, cannot *control* the outcome if they don't know m. That is, if the number of corrupt participants is less than the privacy threshold, then the corrupt participants cannot control the outcome of the reconstruction phase.

Since ghosts are fail-corrupt participants, we require that fail-corrupt participants can't help break privacy or this weaker correctness of the scheme. For privacy, this is true of any verifiable secret sharing scheme (See **Theorem 33**).

Any scheme that satisfies these properties can give us the improved tradeoff. By **Theorems 33** and **41**, ICVSS has the required properties.

5.7.2 Protocol

We use a secret sharing scheme similar to GVSS called IC-GVSS. It is identical to GVSS except that the underlying verifiable secret sharing scheme is ICVSS.

Protocol 3 IC-GHOST-SUM$_{n,s,g}$

Players: n participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Input: $\mathcal{P}_i$ inputs $x_i \in \{0, 1\}$, for $i = 1, \ldots, n$

Output: Each participant gets output $y = x_1 + \cdots + x_n$

Preparation Each $\mathcal{P}_i$ creates s *sets* of $2s$ copies of identical BIN-ARRAY$_n(x_i)$s. The BIN-ARRAYs should vary randomly over different sets, but be identical within a set. $\mathcal{P}_i$ then applies a random permutation to each BIN-ARRAY.

Distribution Each BIN-ARRAY is distributed among all n participants using IC-GVSS-Dist$_{n,g}$. If any call to IC-GVSS-Dist$_{n,g}$ aborts, the sender is excluded from future steps of the protocol.

Verification1 For each set of $2s$ BIN-ARRAYs:

1. Half the BIN-ARRAYs are opened (chosen using RANDOM), including the commitments by the sender of that set. The opened BIN-ARRAYs are checked for well-formedness. If one of a participant's BIN-ARRAYs is found to be ill-formed, his shares are discarded and he is excluded from future steps.
2. For each unopened BIN-ARRAY in this iteration, $\mathcal{P}_i$ broadcasts the permutations on the bins, which each participant applies to his shares of that BIN-ARRAY.

Verification2 For each participant $\mathcal{P}_i$, all s^2 of $\mathcal{P}_i$'s unopened BIN-ARRAYs are put into BIN-ARRAY-EQUALITY. If it returns unequal then $\mathcal{P}_i$'s shares are discarded and he is excluded from future steps.

Computation For each participant $\mathcal{P}_i$, the participants choose the first unopened BIN-ARRAY from each *set* and use these to compute s parallel totals, as in **Protocol 1**. If each repetition does not give the same answer the protocol aborts.

In IC-GHOST-SUM we achieve the desired threshold tradeoff $(t^{(p)}, t^{(c)}, t^{(r)}) = (n - t, n - t, t + 1)$ for any $t \in [0, \frac{n}{2})$, by setting $g = n - 2t$ (**Theorems 42**, **44**, **45** and **46**). Table 1 summarizes the characteristics of the three protocols.

5.7.3 Properties of IC-GVSS

Theorem 37 $t^{(p)}(\text{IC-GVSS}_{n,g}) = \frac{n+g}{2}$.

	$t^{(p)}$	$t^{(c)}$	$t^{(r)}$
BIN-SUM	n	n	1
GHOST-SUM	$n-t$	$t+1$	$t+1$
IC-GHOST-SUM	$n-t$	$n-t$	$t+1$

Table 5.I – Thresholds on privacy $t^{(p)}$, correctness $t^{(c)}$ and robustness $t^{(r)}$, for our three main protocols that compute addition of binary inputs. The parameter $t \in [0, \frac{n}{2})$ yields various tradeoffs for protocols GHOST-SUM and IC-GHOST-SUM. The tradeoff for protocol GHOST-SUM is also applicable to the computation of any linear function.

Proof. This follows from **Theorem 34**. □

Theorem 38 $t^{(r)}(\mathsf{IC\text{-}GVSS}_{n,g}) = \frac{n-g}{2}$.

Proof. This follows from **Theorem 35**. □

Theorem 39 *A set of less than* $\frac{n+g}{2}$ *dishonest receivers cannot forge a signature.That is, they cannot create a share and a signature that convinces an honest participant that the share came from an honest sender.*

Proof. If the sender and a receiver are honest, then the receiver will reject any value different from the intended share except with exponentially small probability by property **SC3**. Therefore, $\frac{n+g}{2} - 1$ corrupt players cannot convince a single honest player to accept the fake share, therefore they cannot get $\frac{n+g}{2}$ participants to accept the fake share, which is required for it to pass. □

Theorem 40 *In* $\mathsf{IC\text{-}GVSS}_{n,g}$*, an honest sender cannot be eliminated in the share phase unless at least* $\frac{n+g}{2}$ *participants are corrupt.*

Proof. A sender can be eliminated in the distribution phase if and only if a participant shows signed shares that are ill-formed and $\frac{n+g}{2}$ participants accept the signature. If a sender is honest, he will give well-formed shares and so any player accepting a signature on an ill-formed share is accepting a forged share, which can't be done by **Theorem 39**. □

Theorem 41 $\mathsf{IC\text{-}GVSS}_{n,g}$ *has property* $\mathbf{C}'$ *when simultaneous broadcast is used in the reconstruction phase.*

Proof. Let $m_1, \ldots, m_n$ be shares distributed using IC-GVSS. Suppose the actively corrupt participants, $\mathcal{P}_1, \ldots, \mathcal{P}_c$ change their shares to $m'_1, \ldots, m'_c$. Let g be a polynomial interpolating the actual shares and g' be a polynomial interpolating the changed shares, $m'_1, \ldots, m'_c, m_{c+1}, \ldots, m_n$. If a simultaneous broadcast channel is used, the new shares cannot be a function of any honest shares.

Consider $(g' - g)(0) = a$. The secret reconstructed from the new shares will be $m' = g'(0) = m + a$. The dishonest coalition knows a, since they know $g' - g$ (the shares of honest parties are all 0). Therefore, they have information about m' if and only if they have information about m. □

5.7.4 Properties of IC-GHOST-SUM

Theorem 42 *In* $t^{(p)}(\text{IC-GHOST-SUM}_{n,g}) = \frac{n+g}{2}$.

Proof. Follows from **Theorem 28**. □

Theorem 43 BIN-ARRAY-EQUALITY *will not output* unequal *for an equal set when there are less than* $\frac{n+g}{2}$ *corrupt participants, except with exponentially small probability.*

Proof. It is not difficult to see that this would require dishonest players to forge shares. This is not possible, except with exponentially small probability, by **Theorem 39**. □

Theorem 44 *If less than* $\frac{n+g}{2}$ *participants are corrupt, then no honest participant will be disqualified except with exponentially small probability.*

Proof. In the distribution phase, an honest participant will not be disqualified except with exponentially small probability, by **Theorem 40**.

In the first verification phase, a participant can only be disqualified if one of his BIN-ARRAYs is opened to an invalid secret. In order to accomplish this, at least one corrupt participant must open an incorrect share with the first participant's signature forged. This cannot be done with less than $\frac{n+g}{2}$ corrupt participants by **Theorem 39**.

In the second verification phase, a participant can only be disqualified if one round of equality outputs unequal. This can't happen to an honest participant except with exponentially small probability, by **Theorem 43**. □

Theorem 45 $t^{(c)}(\text{IC-GHOST-SUM}_{n,g}) = \frac{n+g}{2}$.

Proof. Suppose a dishonest collusion of $\frac{n+g}{2} - 1$ participants want to cause the total Y to be inconsistent with the inputs of honest participants. They must change the value of some shared BIN-ARRAY, X.

By **Theorem 37**, nothing is known about X before the computation (and output) phase. By **Theorem 41**, the dishonest collusion will therefore know nothing about the value of the shared BIN-ARRAY X' that would result from their behaviour. It could be any array of $2n$ integers $\pmod p$. There are p^{2n} of these, and only $2n$ well-formed BIN-ARRAYs. The probability of creating a well-formed BIN-ARRAY is thus less than $\frac{2n}{p^{2n}}$, which is bounded by a constant. Repetition yields exponential security.

□

Theorem 46 *In* $t^{(r)}(\text{IC-GHOST-SUM}_{n,g}) = \frac{n-g}{2}$.

Proof. This proof is identical to that of **Theorem 29**. □

5.8 Applications

The sum of bits is a very basic function that can be useful in several contexts. In this section we present two very distinct applications. The first one, voting, is very natural. The second application is less obvious and regards anonymous transmission of information. We would first like to point out the fact that it is very easy to modify the three SUM protocols to have a more general set of inputs than bits. More precisely, by having each participant simulate k participants, it is possible to perform the sum of integers between 0 and k.

5.8.1 Voting

The multiparty problem of voting is as follows. Players $\mathcal{P}_i$ for $i = 1, \ldots, n$ input $x_i \in \{0, \ldots, c-1\}$. The output is $f(x_1, \ldots, x_n) = (t_1, \ldots, t_c)$ where $t_j = |\{i : x_i = j\}|$. Each player inputs a choice in $\{0, \ldots, c-1\}$ and the output is the tally of how many players voted for each of the c choices.

To see how the same ideas applied to multiparty sum can be applied to voting, we can think of SUM as a vote for one of two choices, where submitting your vote to the tally (i.e., putting your ball in one of the first n bins) corresponds to voting for one choice, and withholding your vote from the tally (i.e., putting your ball in one of the last n bins) corresponds

to voting for the other choice. In fact, we may regard the first n bins as one tally, and the last set of n bins as a separate tally. We can generalise by considering cn bins, with bins $(i-1)n+1,\ldots,in$ corresponding to the ith candidate, or choice. Similar to the protocols for SUM, the total over all bins in the final tally must be equal to n or the protocol aborts.

We can easily extend the voting protocol to allow each voter k votes. This could allow a voter to weigh various candidates as he chooses, for instance, giving several votes to his favourite candidate, who he may feel is unlikely to win, and several votes to his second favourite candidate, who is more likely to win the majority. A further application of multiple votes is a multi-issue ballot, where a voter can divide his votes among issues as he chooses.

Another less straightforward extension of the voting protocol could transform it into a more practical protocol. A set of distinguished players, called voting authorities, can be introduced. Channels would only be required between each participant and each authority, as well as between all authorities. The initial sharing stage would be similar, but all subsequent stages of the protocol would be carried out by the voting authorities. All thresholds now apply to the authorities, including robustness. Note that even if we were to use the simple secret sharing scheme SS, no dishonest coalition of non-authority voters can make the protocol abort. This is true because of the verification stages, which find ill-formed votes except with exponentially small probability.

5.8.2 Anonymous Message Transmission

The multiparty problem of anonymous message transmission is as follows. One or more players $\mathcal{P}_i$ from $i=1,\ldots,n$ inputs a message $x_i \in \{0,1,\perp\}^\ell$ and a designated receiver $\mathcal{R}$ receives the multiset of all transmitted messages $M=\{x_i\}$. We require two privacy properties to hold: if $\mathcal{P}_i$ sends message x_i, then for each $j \neq i$, $\mathcal{P}_j$ learns nothing about x_i; and for each $x_i \in M$, $\mathcal{R}$ has no information about i (the receiver learns nothing about the identity of any sender).

The protocols in [BT07] and [BT08] achieve unconditional privacy, but a single participant can cause the protocol to fail. In addition, the protocols abort whenever two players try to send messages simultaneously. In this section, we describe a protocol which trades privacy for robustness and allows any set of messages to be sent. We do this by slightly modifying the voting protocol from **Section 5.8.1** to achieve anonymous message transmission. We get the same security properties in this modified protocol.

Anonymous bit transmission, the case where $\ell = 1$, can be achieved by a three candidate vote. The first candidate represents 0, the second candidate represents 1, and the third candidate represents no message, $\perp$. We call this a BIT-BIN-ARRAY. At the end of the computation, the shares of the total are sent to the receiver, rather than broadcast to all participants. The receiver then knows how many participants sent him the message 0, and how many sent 1.

We can efficiently generalise the above idea to larger ℓ, by defining a MESSAGE as a vector of ℓ BIT-BIN-ARRAYs. (In the case of a message of arbitrary length, we don't need $\perp$, since we can encode the sending of no message as some string, say the all zeros string, but we leave it in for simplicity). We have two additional requirements.

A bin can be defined by a pair of numbers, (c, b), where $c \in \{0, 1, \perp\}$ represents the candidate, or choice, that the bin falls under, and b denotes the bin number *within that candidate*. The first requirement is that every BIT-BIN-ARRAY in a MESSAGE must have a single non-empty bin, as in a standard vote, but the *bin number* (b) of the non-empty bin must be the same in each message. That is, to show that two bits are part of the same message, they must use the same bin (in different BIT-BIN-ARRAYs).

Secondly, in order to distinguish between different messages, we require that two senders don't choose the same bin number b.

The second requirement can be solved by increasing the number of bins in a BIT-BIN-ARRAY from $3n$ to $3n^2$. We then have $b \in [n^2]$, so in each round, the probability that two messages collide is bounded by a constant. Repetition yields exponential probability that there will be some round in which no two messages collide.

For the first requirement, we simply need to add an extra step to the second verification phase, where we call the following procedure on each MESSAGE for each sender.

Procedure 5 BIN-NUMBER-EQUALITY$_s$

Players: $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Input: $(X^j)_{j=1}^{\ell}$ a set of s LDSs, $X^j = \mathsf{LDS}(\vec{v}_j^{(i)})_{i=1}^{6s}$

Output: equal or unequal

For $k = 1, \ldots, n^2$:

1. For $j = 1, \ldots, \ell$, $Y^j = \vec{v}_j^{(k)} + \vec{v}_j^{(n^2+k)} + \vec{v}_j^{(2n^2+k)}$
2. Compute EQUALITY$(\{Y^j\}_{j=1}^{\ell})$. If it outputs unequal, return unequal.

Return equal.

The above protocol simply ensures that, for any MESSAGE and any bin number $b \in [n^2]$, the sum of the values in bins $(0, b)$, $(1, b)$, and $(\perp, b)$ are same in each BIT-BIN-ARRAY throughout the MESSAGE. If that MESSAGE uses bin b, the sums will all be 1 (since exactly one of 0, 1, and $\perp$ was chosen for each bit of the message) and otherwise the sums will all be 0.

CHAPTER 6

GENERAL CONSTRUCTION OF EFFICIENT MULTIPARTY COMPUTATION

Authors: Bernardo David, Ryo Nishimaki, Samuel Ranellucci, Alain Tapp

6.1 Introduction

Secure multiparty computation (MPC) allows mutually distrustful parties to compute functions on private data that they hold, without revealing their data to each other. Obtaining efficient multiparty computation is a highly sought after goal of cryptography since it can be employed in a multitude of practical applications, such as auctions, electronic voting and privacy preserving data analysis.

Notably, it is known that MPC can be achieved from the garbled circuits technique first proposed by Yao [Yao86] and from a basic primitive called oblivious transfer (OT), which was introduced in [Rab81, EGL85]. The basic one-out-of-two oblivious transfer (OT_1^2) is a two-party primitive where a *sender* inputs two messages m_0, m_1 and a *receiver* inputs a bit c, referred to as the *choice bit*. The receiver learns m_c but not m_{1-c} and the sender learns nothing about the receiver's choice (*i.e.* c). This primitive was proven to be sufficient for achieving MPC in [Kil88, OGW08, CvdGT95].

Even though many approaches for constructing MPC exist, only recently methods that can be efficiently instantiated have been proposed. Among these methods, the IPS compiler [IPS08] stands out as an important construction, achieving MPC without honest majority in the OT-hybrid model. In this work we will focus on the cut-and-choose OT based construction and the improvement of the IPS compiler introduced by Lindell et al. [LP11, LP12, LOP11, Lin13]. Other interesting protocols are introduced in [SS11, DFH12, NNOB12, AJLA+12, LATV12, DPSZ12].

In the approaches for obtaining efficient MPC presented in [LP11, Lin13], the authors employ cut-and-choose OT, where the sender inputs s pairs of messages and receiver can choose to learn both messages b_0, b_1 from $\frac{s}{2}$ input pairs, while he only learns one of the messages in the remaining pairs. A batch version of this primitive is then combined with Yao's protocol

to achieve efficient MPC. In the improvement of the IPS compiler, the authors employ Multi-sender k-out-of-n OT, where j senders input a set of n messages out of which a receiver can choose to receive k messages. These complex primitives are usually constructed from specific number-theoretic and algebraic assumptions yielding little insight to their relationship with other generic and potentially simpler primitives.

In parallel to the efforts for obtaining efficient MPC, research has been devoted to obtaining constructions of basic primitives that can be efficiently combined between themselves in order to obtain more complex primitives and protocols. One of the main approach taken towards this goal has been called *structure preserving cryptography*, which aims at constructing primitives where basically all the public information (*e.g.* signatures, public keys, ciphertexts and commitments) are solely composed of bilinear group elements. This allows for the application of efficient Groth-Sahai non-interactive zero knowledge (NIZK) proof systems [GS08] and efficient compostion of primitives. Until now the main results in this area have been structure preserving signature and commitment schemes [AFG$^+$10, AGHO11] and encryption [CHK$^+$11].

6.1.0.1 Our Contributions:

The central goal of this paper is to present general constructions of the primitives used as the main building blocks in the frameworks of [LP11, LOP11, LP12, Lin13] in the universal composability model [Can01]. We show general relations between such primitives by generically constructing them from simple variants of oblivious transfer. We present three main results:

— **The first *general* constructions of Multi-sender k-out-of-n OT and Batch Single Choice Cut-and-Choose OT:** We show that Multi-sender k-out-of-n OT and Batch Single Choice Cut-and-Choose OT can be obtained from generalized OT (GOT) [IK97] combined with proper access structures. Differently from the original constructions of [LP11, LP12, LOP11, Lin13], our constructions are based on a simple generic primitive and do not require Committed OT or specific computational assumptions. These constructions can be readily used to instantiate the MPC frameworks presented in [LP11, LP12, LOP11, Lin13].

— **Generalized Oblivious Transfer based on Verifiable Oblivious Transfer:** Verifiable Oblivious Transfer (VOT) [CC00, JS07, KSV07] is a flavor of 1-out-of-2 OT

where the sender can reveal one of his messages at any point during the protocol execution allowing the receiver to verify that this message is indeed one of the original sender's inputs. We show that GOT can be obtained from VOT, generalizing even more the constructions describe before. Moreover, our generic construction of GOT may be of independent interest.

— **Structure Preserving Oblivious Transfer and the first *generic* composable constructions of Verifiable Oblivious Transfer:** We introduce structure preserving oblivious transfer (SPOT), which is basically a UC secure 1-out-of-2 OT where the sender's messages and the protocol transcript are represented solely by bilinear group elements. We then build on this characteristic to provide a generic construction of VOT from any SPOT protocol combined with structure preserving extractable or equivocable commitments and Groth-Sahai NIZKs. Differently from the VOT protocols of [CC00, JS07, KSV07], our constructions are totally modular and independent of specific assumptions. Moreover, we introduce a round optimal SPOT protocol based on a protocol by Peikert et al. [PVW08] and observe that the protocols in [CKWZ13] fit our definitions.

Our contributions are two-fold, showing that efficient MPC can be based on a weaker simple primitive (*i.e.* VOT) and providing a generic method for constructing such a primitive. Our results generalize previous approaches for efficient MPC by providing constructions ultimately based on SPOT rather than original Multi Sender k-out-of-n OT and Batch Single Choice Cut-and-Choose OT. Such general constructions help understand the relationship between the these complex variants of oblivious transfer and simpler primitives. In fact, these modular constructions are as efficient as the underlying NIZK proof system, structure preserving commitment and SPOT. Hence, they can easily take advantage of more efficient constructions of these primitives. Furthermore, we introduce the notion of SPOT, which can be useful in other goals apart from general MPC.

6.2 Preliminaries

Notations and Conventions. For any $n \in \mathbb{N} \setminus \{0\}$, let $[n]$ be the set $\{1, \ldots, n\}$. When D is a random variable or distribution, $y \xleftarrow{R} D$ denotes that y is randomly selected from D according to its distribution. If S is a set, then $x \xleftarrow{U} S$ denotes that x is uniformly selected

from S. $y := z$ denotes that y is set, defined or substituted by z. When b is a fixed value, $A(x) \to b$ (e.g., $A(x) \to 1$) denotes the event that machine (or algorithm) A outputs b on input x. We say that function $f : \mathbb{N} \to \mathbb{R}$ is negligible in $\lambda \in \mathbb{N}$ if for every constant $c \in \mathbb{N}$ there exists $k_c \in \mathbb{N}$ such that $f(\lambda) < \lambda^{-c}$ for any $\lambda > k_c$. Hereafter, we use $f < \mathsf{negl}(\lambda)$ to mean that f is negligible in λ. We write $\mathcal{X} \overset{\mathsf{c}}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable.

Bilinear Groups. Let $\mathcal{G}$ be a bilinear group generator that takes security parameter 1^λ as input and outputs a description of bilinear groups $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g})$ where $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ are groups of prime order p, g and $\hat{g}$ are generators in $\mathbb{G}$ and $\mathbb{H}$, respectively, e is an efficient and non-degenerate map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. If $\mathbb{G} = \mathbb{H}$, then we call it the symmetric setting. If $\mathbb{G} \neq \mathbb{H}$ and there is no efficient mapping between the groups, then we call it the asymmetric setting.

Symmetric External Decisional Diffie-Hellman (SXDH) Assumption. Let $\mathcal{G}^{\mathsf{DDH1}}(1^\lambda)$ be an algorithm that on input security parameter λ, generates parameters $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^\lambda)$, chooses exponents $x, y, z \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$, and outputs $\vec{I} := (\Lambda, g^x, g^y)$ and (x, y, z). When an adversary is given $\vec{I} \overset{\mathsf{R}}{\leftarrow} \mathcal{G}_{\mathsf{DDH1}}(1^\lambda)$ and $T \in \mathbb{G}$, it attempts to distinguish whether $T = g^{xy}$ or $T = g^z$. This is called the DDH1 problem. This advantage $\mathsf{Adv}^{\mathsf{DDH1}}_{\mathcal{A}}(\lambda)$ is defined as follows: $\mathsf{Adv}^{\mathsf{DDH1}}_{\mathcal{A}}(\lambda) := \left| \Pr\left[\mathcal{A}(\vec{I}, g^{xy}) \to 1 \middle| (\vec{I}, x, y, z) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}^{\mathsf{DDH1}}(1^\lambda); \right] - \Pr\left[\mathcal{A}(\vec{I}, g^z) \to 1 \middle| (\vec{I}, x, y, z) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}^{\mathsf{DDH1}}(1^\lambda \right.\right.$

Definition 47 (DDH1 assumption) We say that the DDH1 assumption holds if for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH1}}_{\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.

The DDH2 is similarly defined in terms of group $\mathbb{H}$. If both DDH1 and DDH2 assumptions hold simultaneously, then we say that the symmetric external Diffie-Hellman (SXDH) assumption holds.

6.3 Universal Composability

The Universal Composability framework was introduced by Canetti in [Can06] to analyse the security of cryptographic protocols and primitives under arbitrary composition. In this framework, protocol security is analysed by comparing an ideal world execution and a real world execution under the supervision of an *environment* $\mathcal{Z}$, which is represented by a *PPT*

machine and has access to all communication between individual parties. In the ideal world execution, dummy parties (possibly controlled by a *PPT simulator*) interact directly with the ideal functionality $\mathcal{F}$, which works as a fully secure third party that computes the desired function or primitive. In the real world execution, several *PPT* parties (possibly corrupted by a real world adversary $\mathcal{A}$) interact with each other by means of a protocol π that realizes the ideal functionality. The real world execution is represented by the ensemble $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$, while the ideal execution is represented by the $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The rationale behind this framework lies in showing that the environment $\mathcal{Z}$ is not able to efficiently distinguish between $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$ and $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, thus implying that the real world protocol is as secure as the ideal functionality. In this work we consider security against static adversaries, *i.e.* once a party is corrupted it remains so during the whole execution. See [Can01] for further details.

Definition 48 A protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ if, for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that, for every environment $\mathcal{Z}$, the following holds:

$$EXEC_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$$

6.4 Ideal Functionalities

In this section we present the $\mathcal{F}_{CRS}$, $\mathcal{F}_{OT}$and $\mathcal{F}_{COM}$ universally composable ideal functionalities.

COMMITMENT: $\mathcal{F}_{COM}$

— On input `(Commit,`x`)` from the sender records x and send `(committed)` to the receiver.

— On input `(Open)` from the sender, send `(Reveal,`x`)` to the receiver.

OBLIVIOUS TRANSFER: $\mathcal{F}_{OT}$

$\mathcal{F}_{OT}$is parameterized with $N \in \mathbb{N}$ and domain D and interacts with sender **S**, receiver **R**, and an adversary.

— Upon receiving input $(\texttt{Send}, sid, x_0, x_1)$ from $\mathbf{S}$, where $x_0, x_1 \in D$, store (x_0, x_1, sid) and generate a public delayed output $(\texttt{Receipt}, sid)$ to $\mathbf{S}$ and $\mathbf{R}$.

— Upon receiving input $(\texttt{Transfer}, sid, c)$ from $\mathbf{R}$, check if `Transfer` command has been sent with the given sid, if not, send public delayed output $(\texttt{Transfered}, sid, x_c)$ to $\mathbf{R}$, otherwise ignore the command.

COMMON REFERENCE STRING: $\mathcal{F}_{CRS}$ is parameterized by distribution D.

1. On input `(CRS,sid)` from some party P, verify that $sid = (\mathcal{P}, sid')$ where $\mathcal{P}$ is a set of identities, and that $P \in \mathcal{P}$; else ignore the input. If there is no value crs recorded then choose and record value $\mathsf{crs} \xleftarrow{R} D(1^\lambda)$. Send `(CRS,sid,`$\mathsf{crs}$`)` to P.

6.5 Definition of Dual-Mode Cryptosystem

A dual-mode cryptostem is a crypto-sytem where two public keys are derived on key generation. Dual-mode cryptostem can be instantiate under one of two modes, messy and decryption mode. This crypto-system is used to instantiate Oblivious Transfer as shown in [PVW08].

In messy mode, when the receiver generates two keys, one of them will statistically hide any plaintext encrypted under that key and the other can be used to decypher a cyphertext. This is exactly the mode that is useful for Oblivious Transfer. It is possible using the trapdoor of the CRS to determine which is which. This mode allows extraction of receiver's input by the simulator.

In decryption mode, both keys will allow description of the messages. This thus statistically hides the receiver's input and allow extraction of the sender's input by the simulator. Finally, it is important that these two modes are computationally indistinguishable so that the environment cannot distinguish in which model he is part of.

A dual-mode cryptosystem with message space $\{0,1\}^\ell$ consistes of the following PPT algorithms.

$\mathsf{Setup}(1^\lambda, \mu)$**:** On input security parameter λ and mode $\mu \in \{0,1\}$, it outputs CRS crs and trapdoor t. We let $\mathsf{SetupMessy}(\cdot) := \mathsf{Setup}(\cdot, 0)$ and $\mathsf{SetDec}(\cdot) := \mathsf{Setup}(\cdot, 1)$.

$\mathsf{Gen}(\sigma)$**:** On input branch value $\sigma\{0,1\}$, it outputs public key pk and secret key sk on branch σ.

$\mathsf{Enc}(pk, b, m)$**:** On input pk, branch $b \in \{0,1\}$, and message $m \in \{0,1\}^\ell$, it outputs ciphertext c on branch b.

$\mathsf{Dec}(sk, c)$**:** On sk and c, it outputs message m.

$\mathsf{FindMessy}(t, pk)$**:** On input t and pk, it outputs branch value $b \in \{0,1\}$ corresponding to a messy branch of pk.

$\mathsf{TrapGen}(t)$**:** On input t, it outputs public key pk and corresponding secret keys for branches 0 and 1, respectively.

Definition 49 (Dual-Mode Encryption [PVW08]) A dual-mode cryptosystem is a tuple of algorithms described above that satisfy the following properties:

Completeness for decryptable branch: For any $\mu \in \{0,1\}$, any $(\mathsf{crs}, t) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mu)$, any $\sigma \in \{0,1\}$, any $(pk, sk) \xleftarrow{R} \mathsf{Gen}(\sigma)$, and any $m \in \{0,1\}^\ell$, $\mathsf{Dec}(sk, \mathsf{Enc}(pk, \sigma, m)) \to m$.

Indistinguishability of modes: $\mathsf{crs}_0 \overset{c}{\approx} \mathsf{crs}_1$ where $(\mathsf{crs}_\mu, t_\mu) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mu)$.

Trapdoor identification of a messy branch: For any $(\mathsf{crs}, t) \xleftarrow{R} \mathsf{SetupMessy}(1^\lambda)$ and any pk,

$\mathsf{FindMessy}(t, pk) \to b$ such that $\mathsf{Enc}(pk, b, m_0) \overset{s}{\approx} \mathsf{Enc}(pk, b, m_1)$ for any $m_0, m_1 \in \{0,1\}^\ell$.

Trapdoor generation of keys decryptable on both branches: For any $(\mathsf{crs}, t) \xleftarrow{R} \mathsf{SetupDec}(1^\lambda)$, $(k, sk_0, sk_1) \xleftarrow{R} \mathsf{TrapGen}(t)$ such that for any $\sigma \in \{0,1\}$, $(pk, sk_\sigma \overset{s}{\approx} \mathsf{Gen}(\sigma))$.

6.6 Full Description of Groth-Sahai Proof System

Groth and Sahai presented an efficient non-interactive proof system based on bilinear mappings (often called GS-proofs) [GS08]. Their proofs are non-interactive witness indistinguishable (NIWI) or non-interactive zero-knowledge (NIZK) proofs (introduced in [BFM88])

for the satisfiability of equations such as pairing product equations, multi exponentiation equations. GS-proofs are quite useful when witnesses consist of group elements. The GS-proof system can be instantiated under the subgroup decision, SXDH, or decisional linear (DLIN) assumption. The GS-proof system gives efficient non-interactive WI (NIWI) proofs and non-interactive ZK (NIZK) proofs for the satisfiability of equations such as pairing product equations [GS08]. Ghadafi, Smart, and Warinschi corrected some errors in [GS08] at PKC'10 [GSW10]. The GS-proof system can be instantiated under the Subgroup decision, SXDH, and DLIN assumptions.

The Groth-Sahai proof system is based on Groth-Sahai (homomorphic) commitment schemes.

Groth-Sahai Commitment Scheme. In this scheme, we first generate standard parameter $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \xleftarrow{R} \mathcal{G}(1^\lambda)$.

$\mathsf{GS.ComSetup}(\Lambda)$**:** On input Λ, it outputs CRS crs_{com}.

$\mathsf{GS.Commit}(\mathsf{crs}_{com}, m, d)$**:** In order to commit $m \in \mathbb{G}$, it takes as input m, CRS crs_{com}, and decommitment value d and outputs commitment $c \xleftarrow{R} \mathsf{GS.Commit}(\mathsf{crs}_{com}, m, d)$.

$\mathsf{GS.ExpCom}(\mathsf{crs}_{com}, m, b, d)$**:** On input $m \in \mathbb{Z}_p$, base $b \in \mathbb{G}$, CRS crs_{com}, and decommitment d, it outputs (b, c) where $c \xleftarrow{R} \mathsf{GS.Commit}(\mathsf{crs}_{com}, b^m, d)$.

$\mathsf{GS.Opening}(\mathsf{crs}_{com}, c, m, d)$**:** If d is valid decommitment for (m, c), then it outputs 1. Otherwise, it outputs 0. That is, $\mathsf{GS.Opening}(\mathsf{crs}_{com}, c, m, d) \to 1/0$. In the case of verifying that (b, c) is a commitment to exponent m, $\mathsf{GS.Opening}(\mathsf{crs}_{com}, c, b^m, d) \to 1/0$.

The common reference strings of the Groth-Sahai commitment scheme have two types of generation. In one type, the commitment is perfectly binding, computationally hiding, and extractable. In the other type, the commitment is perfectly hiding, computationally binding, and equivocal. These two types are computationally indistinguishable.

The GS commitment scheme based on the SXDH assumption is as follows:

$\mathsf{GS.ComSetup}(1^\lambda)$ It chooses $\alpha, \hat{\alpha}, \beta, \hat{\beta}, \rho, \hat{\rho} \xleftarrow{U} \mathbb{Z}_p$,

Extractable CRS: $\vec{u} := (g, g^\alpha)$, $\vec{v} := (g^\rho, g^{\rho\alpha})$, $xk := \alpha$.

Equivocal CRS: $\vec{u} := (g, g^\alpha)$, $\vec{v} := (g^\rho, g^{\rho\alpha+\beta})$, $ek := (\alpha, \beta)$.

$\mathsf{GS.Commit}(\mathsf{crs}, m, d)$ It chooses $r_1, r_2 \xleftarrow{U} \mathbb{Z}_p$, for $X \in \mathbb{G}$, $\mathsf{Com}(ck, X, r) := (u_1^{r_1} v_1^{r_2}, X \cdot u_2^{r_1} v_2^{r_2}) = (g^{r_1}(g^\rho)^{r_2}, X \cdot (g^\alpha)^{r_1}(g^{\rho\alpha})^{r_2})$.

If the CRS is extractable and we have α, we can extract X like the decryption algorithm of the ElGamal encryption scheme.

Groth-Sahai Proof System. The proof system consists of the following algorithms.

$\mathsf{GS.Setup}(\Lambda)$**:** On input $\Lambda \xleftarrow{R} \mathcal{G}(1^\lambda)$, it outputs CRS GS for the proof system.

$\mathsf{GS.Prove}(GS, S, W)$**:** On input statement S (some equation) and witness W, it outputs non-interactive (WI or ZK) proof of knowledge π.

$\mathsf{GS.Vrfy}(GS, S, \pi)$**:** On input S and π, it verifies the proof and outputs 1 if it is valid, 0 otherwise.

$\mathsf{GS.ExtSetup}(\Lambda)$**:** On input Λ, it outputs CRS GS (this distribution is identical to the normal CRS generated by $\mathsf{GS.Setup}$) and trapdoor td_{ext} for extraction of valid witnesses from valid proofs.

$\mathsf{GS.Extract}(GS, td_{\mathsf{ext}}, \pi)$**:** On input proof π and trapdoor td_{ext}, it extracts witness W that satisfies the statement of π. This extraction does not require rewinding.

$\mathsf{GS.SimSetup}(\Lambda)$**:** It outputs simulated CRS $\widetilde{GS}$ and trapdoor td_{sim}. Simulated CRS $\widetilde{GS}$ is computationally indistinguishable from the normal CRS output by $\mathsf{GS.Setup}(\Lambda)$.

$\mathsf{GS.SimProve}(\widetilde{GS}, td_{\mathsf{sim}}, S)$**:** On input $\widetilde{GS}$ and trapdoor td_{sim}, it outputs simulated proof $\widetilde{\pi}$ for S. It holds that $\mathsf{GS.Vrfy}(\widetilde{GS}, S, \widetilde{\pi}) = 1$. This simulation does not require rewinding.

In the GS-proof system, we can extract witnesses from proofs by using trapdoor td_{ext} as long as those values are *group elements*. Unfortunatly, if we use witness $w \in \mathbb{Z}_p$, then we compute $g^w \in \mathbb{G}$ and generate a proof with witness g^w. Thus, we can only extract g^w. This is not w itself but an output of function $F(w) := g^w$. Therefore, it is said that the GS-proof system has F-extractability [BCKL08].

The statement S to be proven consists of the following list of values: variable $\{X_i\}_i \in \mathbb{G}, \{Y_j\}_j \in \mathbb{H}$ ($i \in [m]$, $j \in [n]$), $\{x_{i'}\}_{i'}, \{y_{j'}\}_{j'} \in \mathbb{Z}_p$ ($i' \in [m']$, $j' \in [n']$), constants $A_i \in \mathbb{G}, B_i \in \mathbb{H}$, $t_T \in \mathbb{G}_T, T_1 \in \mathbb{G}, T_2 \in \mathbb{H}$, $a_i, b_i, \gamma_{i,j}, t \in \mathbb{Z}_p$ as well as commitments $\{C_i\}_i, \{D_j\}_j$ to $\{X_i\}_i \in \mathbb{G}, \{Y_j\}_j \in \mathbb{H}$. Groth and Sahai proposed how to construct non-interactive proofs

of knowledge (NIPK) for the following equations:

$$\begin{aligned}
\prod_{i=1}^{n} e(A_i, Y_i) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(X_i, Y_i)^{\gamma_{i,j}} &= t_T, \\
\prod_{i=1}^{n'} A_i^{y_i} \cdot \prod_{i=1}^{m} X_i^{b_i} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n'} X_i^{\gamma_{i,j} y_j} &= T_1, \\
\prod_{i=1}^{n} Y_i^{a_i} \cdot \prod_{i=1}^{m'} B_i^{x_i} \cdot \prod_{i=1}^{m'}\prod_{j=1}^{n} Y_i^{\gamma_{i,j} x_i} &= T_2, \\
\sum_{i=1}^{n'} a_i y_i + \sum_{i=1}^{m'} x_i b_i + \sum_{i=1}^{m'}\sum_{j=1}^{n'} \gamma_{i,j} x_i y_j &= t.
\end{aligned}$$

This proof π includes commitments $C_1, \ldots, C_m, D_1, \ldots, D_n$. Groth and Sahai provided an efficient extractor that opens these commitments to values $X_1, \ldots, X_m, Y_1, \ldots, Y_n$ that satisfy the pairing product equation.

The proof system satisfies correctness, extractability, CRS indistinguishability, and composable WI (or composable ZK) [GS08].

Correctness: For all $\pi \xleftarrow{\mathsf{R}} \mathsf{GS.Prove}(GS, S, W)$, it holds $\mathsf{GS.Vrfy}(GS, \pi) \to 1$.

Extractability: For $(GS, td_{\mathsf{ext}}) \xleftarrow{\mathsf{R}} \mathsf{GS.ExtSetup}(\Lambda)$ and proof π, if $\mathsf{GS.Vrfy}(GS, \pi) \to 1$, then witness $W := \mathsf{GS.Extract}(GS, td_{\mathsf{ext}}, \pi)$ satisfies statement S with probability 1.

CRS indistinguishability: For $GS \xleftarrow{\mathsf{R}} \mathsf{GS.Setup}(\Lambda)$ and $\widetilde{GS} \xleftarrow{\mathsf{R}} \mathsf{GS.SimSetup}(\Lambda)$, it holds $GS \overset{\mathsf{c}}{\approx} \widetilde{GS}$.

Composable Witness Indistinguishability: For all PPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{cpWI}}_{\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{l} \widetilde{GS} \xleftarrow{\mathsf{R}} \mathsf{GS.SimSetup}(\Lambda); (S, W_0, W_1, s) \xleftarrow{\mathsf{R}} \mathcal{A}(\widetilde{GS}); \\ b \xleftarrow{\mathsf{U}} \{0,1\}; \pi \xleftarrow{\mathsf{R}} \mathsf{GS.Prove}(\widetilde{GS}, S, W_b); b' \xleftarrow{\mathsf{R}} \mathcal{A}(s, \pi) \end{array} \right] - 1 = 0.$$

The GS-proof provides composable zero-knowledge proofs for some restricted class of statements (Σ_{ZK} denotes this class).

Composable Zero Knowledge: For all PPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{cpZK}}_{\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{l} (\widetilde{GS}, td_{\mathsf{sim}}) \xleftarrow{R} \mathsf{GS.SimSetup}(\Lambda); \\ (S \in \Sigma_{ZK}, w, s) \xleftarrow{R} \mathcal{A}(\widetilde{GS}, td_{\mathsf{sim}}); \\ \pi_0 \xleftarrow{R} \mathsf{GS.Prove}(\widetilde{GS}, S, w); \\ \pi_1 \xleftarrow{R} \mathsf{GS.SimProve}(\widetilde{GS}, td_{\mathsf{sim}}, S); \\ b \xleftarrow{U} \{0,1\}; b' \xleftarrow{R} \mathcal{A}(s, \pi_b) \end{array} \right] - 1 = 0.$$

6.7 Dual-Mode Cryptosystem

In this section, we construct a dual-mode cryptosystem which only uses group operations. As a result, we can use the resulting Oblivious Transfer with Groth-Sahai proof systems and convert the Oblivious Transfer into a Verifiable Oblivious Transfer. The idea is that to open one of the message, simply send it and then prove using the Groth Sahai Proof System that that was the message that was sent in the Oblivious Transfer. This will be proven in a later section.

$\mathsf{SetupMessy}(1^\lambda)$ $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \xleftarrow{R} \mathcal{G}(1^\lambda)$, $g_0, g_1 \xleftarrow{U} \mathbb{G}$, $x_0, x_1 \xleftarrow{U} \mathbb{Z}_p$ where $x_0 \neq x_1$. Let $h_b := g_b^{x_b}$ for $b \in \{0,1\}$, $\mathsf{crs} := (g_0, h_0, g_1, h_1)$, and $t := (x_0, x_1)$. It outputs (crs, t).

$\mathsf{SetupDec}(1^\lambda)$ $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \xleftarrow{R} \mathcal{G}(1^\lambda)$, $g_0 \xleftarrow{U} \mathbb{G}$, $y \xleftarrow{U} \mathbb{Z}_p^*$, $g_1 := g_0^y$, $x \xleftarrow{U} \mathbb{Z}_p$, $h_b := g_b^x$ for $b \in \{0,1\}$, $\mathsf{crs} := (g_0, h_0, g_1, h_1)$, and $t := y$. It outputs (crs, t).

$\mathsf{Gen}(\sigma)$ $r \xleftarrow{U} \mathbb{Z}_p$, $g\{g\}^r_\sigma$, $h\{h\}^r_\sigma$, $pk := (g, h) \in \mathbb{G}^2$, $sk := r$. It outputs (pk, sk).

$\mathsf{Enc}(pk, b, m)$ For $pk = (g, h)$ and message $m \in \mathbb{G}$, reads (g_b, h_b) from $\mathsf{crs} = (g_0, h_0, g_1, h_1)$, chooses $s, t \xleftarrow{U} \mathbb{Z}_p$, and computes $u = g_b^s h_b^t$, $v = g^s h^t$. It outputs ciphertext $(u, v \cdot m) \in \mathbb{G}^2$.

$\mathsf{Dec}(sk, c)$ $c = (c_0, c_1)$, It outputs c_1 / c_0^{sk}.

$\mathsf{FindMessy}(t, pk)$ For input $t = (x_0, x_1)$ where $x_0 \neq x_1$, $pk = (g, h)$, if $h \neq g^{x_0}$, then it outputs $b = 0$ as a messy branch. Otherwise, we have $h = g^{x_0} \neq g^{x_1}$, so it outputs $b = 1$ as a messy branch.

$\mathsf{TrapGen}(t)$ For input $t = y$, it chooses $r \xleftarrow{U} \mathbb{Z}_p$, computes $pk := (g_0^r, h_0^r)$ and outputs $(pk, sk := r, sk_1 := r/y)$.

6.7.0.2 Proof Sketch

Theorem 50 *The above construction is a dual-mode cryptosystem.*

In messy mode, CRS is $\mathsf{crs} = (g_0, g_0^{x_0}, g_1, g_1^{x_1})$ and this is non-DDH tuple. In decryption mode, CRS is $\mathsf{crs} = (g_0, g_0^x, g_1, g_1^x)$ and this is DDH tuple. These CRSs are computationally indistinguishable by the SXDH assumption. In messy mode, public key is $pk = (g_\sigma^r, h_\sigma^r) = (g_\sigma^r, g_\sigma^{x\sigma r})$ for $x_0 \neq x_1$, so if we have trapdoor $t = (x_0, x_1)$, we can test $h = g^{x_0}$ or not. If $\sigma = 1$, then it holds $h \neq g^{x_0}$ and $b = 0$ is the messy branch. Otherwise, $b = 1$ is the messy branch.

In decryption mode, public and secret keys are $(pk, sk_0, sk_1) = ((g_0^r, h_0^r), r, r/y)$, so (pk, sk_0) is identically distributed to $\mathsf{Gen}(0)$. If we set $r := r'y$, then we can rewrite $(pk, sk_1) = (g_0^{r'y}, h_0^{r'y}, r') = (g_1^{r'}, h_1^{r'}, r')$ and this is identically distributed to $\mathsf{Gen}(1)$ since $r' = r/y \in \mathbb{Z}_p$ is uniformly random.

6.7.1 UC OT based on Dual-mode Cryptosystem [PVW08]

Let $\mathsf{mode} \in \{\mathsf{mes}, \mathsf{dec}\}$. Peikert, Water, and Vaikuntanathan proposed protocol $\mathsf{dm}^{\mathsf{mode}}$ as follows: Parties use the dual-mode cryptosystem described above. Sender $\mathbf{S}$ has input $m_0, m_1 \in \mathbb{G}$, receiver $\mathbf{R}$ has input $\sigma \in \{0,1\}$. $\mathbf{S}$ and $\mathbf{R}$ query $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$ with $(sid, \mathbf{S}, \mathbf{R})$ and gets (sid, crs). If mode is $\mathsf{mes}/\mathsf{dec}$, then crs is generated by $\mathsf{SetupMessy}/\mathsf{SetupDec}$. First, $\mathbf{R}$ computes $(pk, sk) \xleftarrow{\mathsf{R}} \mathsf{Gen}(\mathsf{crs}, \sigma)$, sends $(sid, ssid, pk)$ to $\mathbf{S}$. When $\mathbf{S}$ receives $(sid, ssid, pk)$, it computes $y_b \xleftarrow{\mathsf{R}} \mathsf{Enc}(pk, b, m_b)$ for each $b \in \{0,1\}$, and sends $(sid, ssid, y_0, y_1)$ to $\mathbf{R}$. When $\mathbf{R}$ receives $(sid, ssid, y_0, y_1)$, it outputs $(sid, ssid, \mathsf{Dec}(sk, y_\sigma))$.

Theorem 51 ([PVW08]) *Let* $\mathsf{mode} \in \{\mathsf{mes}, \mathsf{dec}\}$. *If the SXDH assumption holds, protocol* $\mathsf{dm}^{\mathsf{mode}}$ *securely realizes* $\hat{\mathcal{F}}_{OT}$ *in the* $\mathcal{F}_{CRS}^{\mathsf{mode}}$*-hybrid mode in the static corruption model.*

The messages exchanged in protocol $\mathsf{dm}^{\mathsf{mode}}$ are all group elements, so it is compatible with GS-proofs. The following commitment scheme based on the SXDH assumption is used to construct GS-proofs based on the SXDH assumption. The CRS is $\mathsf{crs} := (\vec{u} := (u_1, u_2), \vec{v} := (v_1, v_2))$ where $\vec{u}, \vec{v} \in \mathbb{G}^2$, the witness is $X \in \mathbb{G}$, and the commitment is $\mathsf{Com} := (u_1^{r_1} v_1^{r_2}, X \cdot u_2^{r_1} v_2^{r_2})$ where $r_1, r_2 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$.

Thus, in protocol $\mathsf{dm}^{\mathsf{mode}}$, if the sender generates a commitment of m_b by the above commitment scheme, then it can prove that the message inside encryption $y_b = \mathsf{Enc}(pk, b, m_b)$

is consistent with the message inside the commitment by GS-proofs. That is, for $(c_0, c_1) = (u, v \cdot m_b) = (g_b^s h_b^t, g^s h^t \cdot m_b)$ and $\mathsf{Com} := (com_1, com_2) = (u_1^{r_1} v_1^{r_2}, u_2^{r_1} v_2^{r_2} \cdot m_b)$, it should hold $c_1 / com_2 = g^s h^t u_2^{-r_1} v_2^{-r_2}$. Equivalently,

$$\begin{pmatrix} g_b & h_b & 1 & 1 \\ 1 & 1 & u_1 & v_1 \\ g & h & u_2^{-1} & v_2^{-1} \end{pmatrix} \cdot \begin{pmatrix} s \\ t \\ r_1 \\ r_2 \end{pmatrix} := \begin{pmatrix} g_b^s \cdot h_b^t \cdot 1^{r_1} \cdot 1^{r_2} \\ 1^s \cdot 1^t \cdot u_1^{r_1} \cdot v_1^{r_2} \\ g^s \cdot h^t \cdot (u_2^{-1})^{r_1} (v_2^{-1})^{r_2} \end{pmatrix} = \begin{pmatrix} g_b^s h_b^t \\ u_1^{r_1} v_1^{r_2} \\ g^s h^t u_2^{-r_1} v_2^{-r_2} \end{pmatrix}.$$

Thus, we can use Groth-Sahai NIZK proofs for the linear equation above.

6.7.2 NIZK for Linear Equation

In order to prove that revealed m_b is consistent with the messages exchanged in the protocol execution, we use Groth-Sahai NIZK proofs. We borrow the notation of Dodis, Haralambiev, López-Alt, and Wichs [DHLW10]. In the UC OT protocol described above, the sender prove that the message inside encryption $y_b = \mathsf{Enc}(pk, b, m_b)$ is the same m_b revealed by the sender. That is, it should hold that for $(c_0, c_1) = (u, v \cdot m_b) = (g_b^s h_b^t, g^s h^t \cdot m_b)$ and $\mathsf{GS.Commit}(gk, m_b, (r_1, r_2)) = (com_1, com_2) = (u_1^{r_1} v_1^{r_2}, u_2^{r_1} v_2^{r_2} \cdot m_b)$, $c_1 / com_2 = g^s h^t u_2^{-r_1} v_2^{-r_2}$, that is

$$\begin{pmatrix} g_b & h_b & 1 & 1 \\ 1 & 1 & u_1 & v_1 \\ g & h & u_2^{-1} & v_2^{-1} \end{pmatrix} \begin{pmatrix} s \\ t \\ r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} g_b^s h_b^t \\ u_1^{r_1} v_1^{r_2} \\ g^s h^t u_2^{-r_1} v_2^{-r_2} \end{pmatrix}.$$

Thus, we use Groth-Sahai NIZK proofs for the equation above. The matrix multiplication of $\boldsymbol{A} \in \mathbb{G}^{m \times n}$ and $X \in \mathbb{Z}^{n \times k}$,

$$\boldsymbol{A} = \{\boldsymbol{a}_{i,j}\}_{m \times n} = \begin{pmatrix} — & \vec{\boldsymbol{a}}_1^\top & — \\ & \cdots & \\ — & \vec{\boldsymbol{a}}_m^\top & — \end{pmatrix}, \; X = \{x_{i,j}\}_{n \times k} = \begin{pmatrix} | & & | \\ \vec{x}_1 & \cdots & \vec{x}_k \\ | & & | \end{pmatrix}$$

is given by $\boldsymbol{A} \cdot X := \{\boldsymbol{b}_{i,j}\}_{m \times k}$ where $:= \langle \vec{\boldsymbol{a}}_i, \vec{x}_j \rangle = \prod_{r=1}^{n} \boldsymbol{a}_{i,j}^{x_{r,j}}$. For $\boldsymbol{\Upsilon} = \{\boldsymbol{\gamma}_{i,j}\}_{n \times k} \in \mathbb{H}^{n \times k}$, then $\boldsymbol{A} \bullet \boldsymbol{\Upsilon} := \{\boldsymbol{t}_{i,j}\}_{n \times k} \in \mathbb{G}_T^{n \times k}$ where $\boldsymbol{t}_{i,j} = \prod_{r=1}^{n} e(\boldsymbol{a}_{i,r}, \boldsymbol{\gamma}_{r,j})$. Dodis, et al [DHLW10]

expressed GS NIZK for linear equations in a general form as follows.

$$\begin{pmatrix} \boldsymbol{b}_{1,1} & \boldsymbol{b}_{1,2} & \cdots & \boldsymbol{b}_{1,N} \\ \boldsymbol{b}_{2,1} & \boldsymbol{b}_{2,2} & \cdots & \boldsymbol{b}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b}_{M,1} & \boldsymbol{b}_{M,2} & \cdots & \boldsymbol{b}_{N,M} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_M \end{pmatrix}$$

Group description is as follows: $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \gamma_0) \xleftarrow{\mathsf{R}} \mathcal{G}(1^\lambda)$. If we set $K = 1$, then the proof is instantiated with the SXDH assumption in asymmetric groups.

GS.Setup: Outputs CRS $\mathsf{crs} := \boldsymbol{\Upsilon}$ and $tk := \vec{t}$ where $\gamma_1, \ldots, \gamma_K \xleftarrow{\mathsf{U}} \mathbb{H}$, $\vec{t} := (t_1, \ldots, t_K) \xleftarrow{\mathsf{U}} \mathbb{Z}_p^K$, and

$$\boldsymbol{\Upsilon} = \begin{pmatrix} \gamma_0^{\sum_{i=1}^K t_i} & \gamma_1^{t_1} & \gamma_2^{t_2} & \cdots & \gamma_K^{t_K} \\ \gamma_0 & \gamma_1 & 1 & \cdots & 1 \\ \gamma_0 & 1 & \gamma_2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_0 & 1 & 1 & \cdots & \gamma_K \end{pmatrix} \in \mathbb{H}^{(K+1)\times(K+1)}.$$

GS.Prove: For inputs $\boldsymbol{B} \in \mathbb{G}^{M\times N}, \vec{\boldsymbol{c}} \in \mathbb{G}^M, \vec{w} \in \mathbb{Z}_p^N$, it chooses $R \xleftarrow{\mathsf{U}} \mathbb{Z}_p^{N\times K}$ and outputs $\pi := (\boldsymbol{\Delta}, \boldsymbol{P})$ where

$$\boldsymbol{\Delta} := \left(\begin{array}{c|c} w_1 & \\ \vdots & R \\ w_N & \end{array}\right) \cdot \boldsymbol{\Upsilon},\ \boldsymbol{P} := \boldsymbol{B} \cdot R$$

for $\boldsymbol{\Delta} \in \mathbb{H}^{N\times(K+1)}$, $\boldsymbol{P} \in \mathbb{G}^{M\times K}$.

GS.SimProve: For inputs $\boldsymbol{B} \in \mathbb{G}^{M\times N}, \vec{\boldsymbol{c}} \in \mathbb{G}^M$, it chooses $R \xleftarrow{\mathsf{U}} \mathbb{Z}_p^{N\times K}$ and outputs $\pi := (\boldsymbol{\Delta}, \boldsymbol{P})$ where

$$\boldsymbol{\Delta} := \left(\begin{array}{c|c} 0 & \\ \vdots & R \\ 0 & \end{array}\right) \cdot \boldsymbol{\Upsilon},\ \boldsymbol{P} := \left(\begin{array}{c|c} \boldsymbol{c}_1 & \\ \vdots & \boldsymbol{B} \\ \boldsymbol{c}_M & \end{array}\right) \cdot \left(\begin{array}{c} -t_1 \ \cdots \ -t_N \\ \hline R \end{array}\right)$$

where $\boldsymbol{\Delta} \in \mathbb{H}^{N\times(K+1)}$, $\boldsymbol{P} \in \mathbb{G}^{M\times K}$.

GS.Vrfy: For input $\pi = (\boldsymbol{\Delta}, \boldsymbol{P})$, it outputs 1 iff it holds that

$$\boldsymbol{B} \bullet \boldsymbol{\Delta} \;:=\; \left(\begin{array}{c|c} \boldsymbol{c}_1 & \\ \vdots & \boldsymbol{P} \\ \boldsymbol{c}_M & \end{array} \right) \bullet \boldsymbol{\Upsilon}.$$

6.8 Generic Construction of Verifiable OT from Structure Preserving OT

In this section, we introduce Structure Preserving Oblivious Transfer (SPOT) and use it to construct verifiable oblivious transfer (VOT) . SPOT will be the basic building block of our efficient general constructions of secure multiparty computation. It is first used in a general transformation that yields VOT, which will be used to obtain generalized oblivious transfer (GOT) and more complex primitives later on.

6.8.1 Structure Preserving Oblivious Transfer

Basically we require all the SPOT protocol messages (*i.e.* the protocol transcript) and the sender's input messages in the protocol to be composed solely of group elements, this allows us to apply GS proofs to prove relations between the parties' inputs and the protocol transcript. Further on, our general transformation will rely on GS proofs to show that a given sender input is associated with a specific protocol transcript.

Definition 52 (Structure Preserving Oblivious Transfer) Formally a structure preserving oblivious transfer protocol defined over a bilinear group $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g})$ must have the following properties:

1. Each of the sender's input messages m_0, m_1 consists of an element of $\mathbb{G}$ or $\mathbb{H}$.
2. All the messages exchanged between **S** and **R** (*i.e.* the protocol transcript) consist of elements of $\mathbb{G}$ and $\mathbb{H}$.
3. The relation between a given input message m_b, $b \in \{0, 1\}$ and a given protocol transcript is expressed by a set of pairing product equations or multi exponentiation equations.

Notice that our general transformations can be applied to any OT protocol in a bit by bit approach, by mapping the binary representation of each element in a given protocol to specific group elements representing 0 and 1 and applying GS proofs individually to each of those elements. However, this trivial approach is extremely inefficient, the number of GS proofs and group elements exchanged between parties would grow polynomially. The first OT protocol to fit this definition was proposed in [GH08], but it relies simultaneously on SXDH, DLIN and q-hidden LSRW assumptions. A recent result by Choi *et. al.* [CKWZ13] also introduced OT protocols based on DLIN and SXDH that match out definition of SPOT. However, these protocols already require a GS proof themselves, introducing extra overhead in applications that combine SPOT with GS proofs.

6.8.2 Obtaining SPOT from Dual-Mode Cryptosystems

Peikert, Vaikuntanathan, and Waters proposed a general framework for constructing universally composable oblivious transfer protocols by using dual-mode encryption schemes [PVW08]. They proposed dual-mode encryption schemes based on the DDH, QR, LWE assumptions [PVW08]. The DDH-base scheme can be instantiated in asymmetric pairing groups under the SXDH assumption. We construct such a dual-mode cryptosystem in Appendix 6.7. Note that the CRS, all messages which are exchanged between the sender and the receiver, and inputs to the sender are all group elements. Therefore, the composable OT protocol of Peikert et al. based on the dual mode cryptosystem presented in Appendix 6.7 is a SPOT protocol compatible with GS-proofs based on the SXDH assumption.

6.8.3 Obtaining VOT

Verifiable oblivious transfer is basically an 1-out-of-2 oblivious transfer where the sender may choose to open one of its input messages m_b where $b \in \{0, 1\}$ at any time, in such a way that the receiver is able to verify that this message had indeed been provided as input. This notion is formalized by the following ideal functionality:

Verifiable Oblivious Transfer: $\mathcal{F}_{VOT}$

— Upon receiving `(Send,`x_0`,`x_1`,sid)` from the sender, if sid has not already been used, store (x_0, x_1, sid) and send `(Receipt,sid)` to the sender and the receiver.

— Upon receiving `(Transfer,c,sid)` from the receiver, check if `Transfer` command has been sent with the given sid, if not, send `(transfered,sid,`x_c`)` to the receiver, otherwise ignore the command.
— Upon receiving `(Open,b,sid)` from the sender, send `(reveal,b,`x_b`,sid)` to the receiver.

We will construct a general protocol π_{VOT} that realizes $\mathcal{F}_{VOT}$ from any universally composable SPOT protocol π_{SPOT} by combining it with a structure preserving commitment π_{Com} (such as the schemes in [GS08][AFG+10]) and Groth-Sahai NIZK proofs. We consider that both parties are running the underlying universally composable structure preserving oblivious transfer protocol $SPOT$ and describe the extra steps needed to obtain VOT. An interesting property of this generic protocol is that even though it was designed for an underlying structure preserving protocol that realizes the 1-out-of-2 OT functionality $\mathcal{F}_{OT}$, it can be applied multiple times to the individual transfers of an adaptive OT protocol in order to obtain verifiable adaptive OT. In this case, the same CRS can be reused for all the individual transfers. Notice that this is the first generic construction of universally composable VOT.

6.8.3.1 Protocol π_{VOT}:

S inputs two messages m_0, m_1 and **R** inputs a choice bit c.

— **Setup:** A common reference string is generated containing the following information:
  — The public parameters for a sound instance of a Groth-Sahai non-interactive zero knowledge proof system.
  — The CRS for the underlying structure preserving commitment scheme π_{Com}.
  — The CRS for the underlying UC structure preserving OT π_{SPOT}.
— **Commitment phase:** Before starting π_{SPOT}, **S** commits to m_0 and m_1 by sending $(sid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ to **R**, where $m_0, m_1 \in_R \{0,1\}^n$ (Notice that it is possible to efficiently map the messages into corresponding group elements that will serve as inputs to π_{SPOT} [GH08]).
— π_{SPOT} **protocol execution S** and **R** run π_{SPOT} storing all the messages exchanged during the protocol execution up to the end of π_{SPOT} with **S**'s input (m_0, m_1) and **R**'s input c or until **S** decides to reveal one of its messages.

— **Reveal phase:** If **S** decides to reveal one of its messages m_b where $b \in \{0,1\}$ at any point of the protocol execution it sends a decommitment to m_b and a GS-proof ψ that the messages exchanged up to that point of the execution contain a valid transfer of message m_b (this proof is zeo-knowledge), sending $(sid, b, \mathsf{Open}(\mathsf{m_b}), \psi)$ to **R**.

— **Verification phase:** After receiving the decommitment and the GS-proof, **R** verifies ψ and the decommitment validity. If both are valid, it accepts the revealed bit, otherwise it detects that **S** is cheating. If the protocol π_{SPOT} did not reach its end yet, **S** and **R** continue by executing the next steps, otherwise they halt.

Theorem 1 *For every universally composable structure preserving oblivious transfer protocol π_{SPOT} and every universally composable structure preserving commitment scheme π_{Com}, Protocol π_{VOT} securely realizes the functionality F_{VOT} in the $\mathcal{F}_{CRS}$-hybrid model.*

Before proceeding to the security proof we show that the protocol works correctly. First of all, notice that since π_{SPOT} is a structure preserving oblivious transfer protocol it is possible to prove statements about the sender's input messages and the protocol transcript using Groth-Sahai NIZK proof systems. Correctness of Protocol π_{VOT} in the case that no Reveal phase happens follows from the correctness of protocol π_{SPOT}. The correctness of the Reveal phase follows from the commitment scheme's security and the GS-proof completeness and soundness. When **S** opens the commitment, **R** is able to check whether the revealed message is indeed one of the messages that **S** used as input in the beginning of the protocol and by verifying the GS-proof, **R** is able to check that the input message m_b is contained in the messages exchanged by both parties meaning that this message is indeed used in the protocol execution. The full proof is presented in Appendix 6.9

6.9 Security Proof of π_{VOT}

In order to prove the security of this protocol we construct simulator that interacts with an internal copy of the adversary $\mathcal{A}$ that may corrupt parties, the ideal functionality $\mathcal{F}_{VOT}$ and an environment Z. For the sake of simplicity, we present the case of a corrupted **S** and that of a corrupted **R** separately. In the trivial case where both parties are corrupted, the simulator $\mathcal{S}$ simply forwards all the messages between Z and $\mathcal{A}$. Analogously, when both parties are honest, $\mathcal{S}$ forwards all the messages between the parties **S**, **R** and Z.

In order to construct the simulators we will use the following setup:

Setup: Simulator $\mathcal{S}$ generates a common reference string by computing the following information:

— The "simulated" public parameters for an of a Groth-Sahai NIZK proof system.

— The "simulated" CRS for the underlying structure preserving commitment scheme π_{Com} that allows the simulator to generate an equivocal commitment using trapdoor t.

— The "simulated" CRS for the underlying UC structure preserving OT π_{SPOT} that allows the simulator to obtain the receiver's choice bit c or the sender's input messages m_0, m_1.

6.9.1 Simulator for the Case of a corrupted S:

Simulator $\mathcal{S}$ interacts with a corrupted sender $\mathcal{A}$, the ideal functionality $\mathcal{F}_{VOT}$ and the environment Z.

— **Setup:** When $\mathcal{A}$ requests a CRS, $\mathcal{S}$ responds by sending the CRS described above.

— **Commitment Phase:** $\mathcal{S}$ waits for the commitments $(sid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ from $\mathcal{A}$ and then interacts with it using protocol π_{SPOT}.

— π_{SPOT} **protocol execution** $\mathcal{S}$ and $\mathcal{A}$ run π_{SPOT} storing all the messages exchanged during the protocol execution up to the end of π_{SPOT} or until $\mathcal{A}$ decides to reveal one of its messages. If $\mathcal{A}$ doesn't decide to reveal one of its messages before the protocol ends, $\mathcal{S}$ uses the same procedures of the original π_{SPOT} simulator to extract m_0, m_1 and sends $(\texttt{Send}, m_0, m_1, \texttt{sid})$ to $\mathcal{F}_{VOT}$.

— **Reveal phase:** If the $\mathcal{A}$ decides to reveal one of its messages m_v at any point of the protocol execution sending $\mathcal{S}$ the message $(sid, b, \mathsf{Open}(\mathsf{m_b}), \psi)$, $\mathcal{S}$ verifies whether the the revealed message is correct by using the same procedures of a honest receiver. If it is correct, $\mathcal{S}$ sends (Open, b, sid) to $\mathcal{F}_{VOT}$. Otherwise, it reveals an invalid bit by sending a corrupted message to $\mathcal{F}_{VOT}$. $\mathcal{S}$ then proceeds by simulating the execution with $\mathcal{A}$ using the same procedures of the original π_{SPOT} simulator.

Lemma 1 *When $\mathcal{A}$ corrupts only the sender, for any universally composable structure preserving oblivious transfer protocol π_{SPOT} and structure preserving commitment scheme π_{Com}, the following holds:*

$$EXEC_{\pi_{VOT},\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} IDEAL_{\hat{\mathcal{F}}_{VOT},\mathcal{S},\mathcal{Z}}$$

Proof. Note that the simulator $\mathcal{S}$ generates all messages exactly as the real protocol and as the original simulator for protocol π_{SPOT}. Thus, the simulation is indistinguishable from the real execution with protocol π_{VOT}. The difference in the CRS is computational indistinguishable since the simulated common reference strings from the commitment protocol, since the GS-proof system and the structure preserving OT protocol are computationally indistinguishable from the real common reference string used for each of these protocols. □

6.9.2 Simulator for the Case of a corrupted receiver:

Simulator $\mathcal{S}$ interacts with a corrupted sender $\mathcal{A}$, ideal functionality $\mathcal{F}_{VOT}$ and environment Z.

— **Setup:** When $\mathcal{A}$ requests a CRS, $\mathcal{S}$ responds by sending the CRS described above.

— **Commitment phase:** Before starting π_{SPOT}, $\mathcal{S}$ commits to two random messages $m_0, m_1 \in_R \{0,1\}^n$ by sending $(sid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ to $\mathcal{A}$.

— π_{SPOT} **protocol execution** $\mathcal{S}$ and $\mathcal{A}$ run π_{SPOT} storing all the messages exchanged during the protocol execution up to the end of π_{SPOT} or until $\mathcal{S}$ receives a message $(\mathsf{Reveal}, b, m'_b), sid$ from $\mathcal{F}_{VOT}$. If not interrupted, $\mathcal{S}$ follows the procedures of the original π_{SPOT} simulator, extracts the choice bit c and sends `(Transfer,c,sid)` to $\mathcal{F}_{VOT}$, obtains m_c and sends it to $\mathcal{A}$ to complete the protocol.

— **Reveal phase:** If $\mathcal{S}$ receives a message $(\mathsf{Reveal}, b, m'_b, sid)$ from $\mathcal{F}_{VOT}$ (meaning that the honest sender revealed his message m_b) it reveals the same bit m_b to $\mathcal{A}$ by doing the following:

1. $\mathcal{S}$ generates an arbitrary opening to $\mathsf{Open}(\mathsf{m}'_\mathsf{b})$ corresponding to the original commitment $\mathsf{Com}(m_b)$ using the trapdoor corresponding to the simulated CRS for π_{Com}.
2. $\mathcal{S}$ generates a proof ψ that the messages exchanged with $\mathcal{A}$ contain a valid transfer of m'_b using the simulated CRS for the GS NIZK proof system.
3. $\mathcal{S}$ sends $(sid, b, \mathsf{Open}(\mathsf{m}'_\mathsf{b}), \psi)$ to $\mathcal{A}$.

— If there are still steps of π_{SPOT} to be executed, $\mathcal{S}$ proceeds as the original π_{SPOT} simulator until the end of the protocol.

Lemma 2 *When $\mathcal{A}$ corrupts only the receiver, for any universally composable structure preserving oblivious transfer protocol π_{SPOT} and any universally composable structure preserving*

commitment scheme π_{Com} the following holds:

$$EXEC_{\pi_{VOT},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\hat{\mathcal{F}}_{VOT},\mathcal{S},\mathcal{Z}}$$

Proof. The only deviations from the real protocol π_{VOT} lie in the commitments $(sid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ sent in the beginning of the protocol and in the case that the honest sender decides to reveal one of its messages. In contrast to the real protocol, $\mathcal{S}$ commits to random $m_0, m_1 \in_R \{0,1\}^n$, and computational indistinguishability from commitments to the real messages follows from the commitments hiding property.

When the honest sender decides to reveal one of the messages, $\mathcal{S}$ deviates from the real protocol by opening one of the previously sent commitments to an arbitrary value and generates a GS-proof of a relation for which it does not know the witnesses. The underlying commitment protocol has been set up with a simulated CRS. This allows the simulator who knows the trapdoor t to issue decommitments to arbitrary values and this difference is computationally indistinguishable. The indistinguishability for the GS-proof that the messages exchanged between the parties contain a transfer where $m_b = m'_b$ follows from the zero knowledge property of GS-proof systems that are set up with the simulated CRS. The only remaining difference is the CRS, which is indistinguishable since the simulated common reference strings of the underlying protocols are themselves computationally indistinguishable from the real common reference strings. □

6.10 Generalized Oblivious Transfer

Generalized Oblivious Transfer is an interesting application of Verifiable Oblivious Transfer. An interesting way of describing an OT is by describing the groups of messages that the receiver can get as sets in a collection. In the case of a simple OT, he can learn one of the sets in $\{\{1\},\{2\}\}$. The k-out-of-n OT is an OT with a collection that contains all the sets of messages of size k or less. This mindset allows us to generalize oblivious transfer to its most general form. There is an important link between generalized oblivious transfer and general access structures.

Definition 2

— *Let $I = \{1, 2, ..., n\}$ be a set of indices. A collection $\mathcal{A} \subseteq \mathcal{P}(I)$ is* **monotone** *if the*

fact that $\mathcal{B} \in \mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{C}$ implies that $\mathcal{C} \in \mathcal{A}$.

— *An* **access structure** *is a monotone collection $\mathcal{A}$ of non-empty sets of I. A set s is* **authorized** *if $s \in \mathcal{A}$ and a set s' is* **minimal** *if there exists no strict subset s'' of s' such that $s'' \in \mathcal{A}$.*

— *The* **complement** *of a collection $\mathcal{C}$ is defined as $\mathcal{C}^* = \{b \subseteq I \mid \exists\, c \in \mathcal{C},\ b = I - c\}$.*

— *We define* **Closure**$(\mathcal{C}) = \{c \subseteq c' \mid c' \in \mathcal{C}\}$.

— *A collection $\mathcal{C}$ is* **enclosed** *if $\mathcal{C} =$***Closure**$(\mathcal{C})$.

— *An element $c \in \mathcal{C}$ is* **maximal** *if there exists no $c' \in \mathcal{C}$ such that $c \subseteq c'$ and $c \neq c'$.*

Theorem 53 *For every enclosed collection C, there exists a unique access structure $\mathcal{A}$ such that $\mathcal{C}^* = \mathcal{A}$*

See [SSR08] for a full proof.

Definition 3 *A* **secret sharing scheme** *is a triplet of randomized algorithms (Share, Reconstruct, Check) over a domain D with an access structure $\mathcal{A}$. $Share_{\mathcal{A}}(s)$ always output $(s_1, \ldots, s_n)$ such that:*
(1) for all $A \in \mathcal{A}$, $Reconstruct_{\mathcal{A}}(\{(i, s_i) \mid i \in A\}) = s$,
(2) for any $A' \notin \mathcal{A}$, $\{(i, s_i) \mid i \in A'\}$ gives no information about s.
$Check_{\mathcal{A}}(s_i) = 1$ if and only if for all $A \in \mathcal{A}$, $Reconstruct_{\mathcal{A}}(\{(i, s_i) \mid i \in A\}) = s$.

Definition 4 *Shares $\{s_i\}$ are* **consistent** *if for any authorized subset of shares, the reconstructed secret is the same.*

GENERALIZED OBLIVIOUS TRANSFER: $\mathcal{F}_{GOT}$

Let $\mathcal{I}$ be an enclosed collection and $m_1, \ldots, m_n \in \{0,1\}^l$.

— Upon receiving `(Send,` $m_1, ..., m_n$`)` from the sender, if such a command was not done previously, store $(m_1, ..., m_n)$.

— Upon receiving `(Choice,I)` where $I \in \mathcal{I}$ is a set of indices, if no `Choice` command was previously sent, and I is in $\mathcal{I}$, then for each $i \in I$, send `(Reveal,i,`m_i`)` to receiver.

6.10.1 Protocol

In this section, we will give a protocol that implements $\mathcal{F}_{GOT}$ in the $\mathcal{F}_{VOT}$, $\mathcal{F}_{COM}$ hybrid model with the aid of secret sharing. The fact that every enclosed collection is the complement of an access structure will be key to this construction. The protocol requires n instances of $\mathcal{F}_{VOT}$. The size of the elements transferred in the $\mathcal{F}_{VOT}$ is the maximum between the length of the messages and the size of the shares which depends on the underlying access structure. The probability that the receiver breaks the protocol is D^{-1} where D is the number of possible element of the secret sharing use in the protocol. Let s be a security parameter and $\mathcal{I}$ be an enclosed collection that defines the subset of messages that are accessible to the receiver.

Protocol: $\pi_{GOT(\mathcal{I})}$

1. The sender selects $k_1, k_2, ..., k_n \xleftarrow{U} \{0,1\}^l$ (one-time pads)
2. Let $\mathcal{A} = \mathcal{I}^*$, $s \xleftarrow{U} D$ and $(s_1, s_2, ..., s_n) = Share_{\mathcal{A}}(s)$.
3. The sender selects a set of n ids that have never been used, denote these ids as sid_i and sends them to the receiver. For each i, sender sends $(\texttt{send}, k_i, s_i, \text{sid}_\text{i})$ to $\mathcal{F}_{VOT}$.
4. Let $I \in \mathcal{I}$ be the set of messages that the receiver wishes to receive, he sets $b_i = 0$ when $i \in I$ otherwise he sets $b_i = 1$. For each i, the receiver sends $(\texttt{Transfer}, b_i, sid_i)$ to $\mathcal{F}_{VOT}$ and records the result.
5. The receiver executes the recover algorithm with the shares he received and obtains S. If the reconstruction failed, he chooses an arbitrary value for S instead. The receiver sends $(\texttt{commit}, \texttt{S})$ to $\mathcal{F}_{COM}$.
6. For each i, the sender sends $(\texttt{open}, 1, sid_i)$ to $\mathcal{F}_{VOT}$.
7. The receiver abort if $\text{Check}_{\mathcal{A}}(s_1, s_2, ..., s_n) \neq 1$.
8. The receiver sends open to $\mathcal{F}_{COM}$. The sender verifies that $S = s$ and if not, aborts the protocol.
9. The sender sends $z_i = m_i \oplus k_i$ to the receiver. ($\{m_i\}$ are the messages)

Theorem 54 *π_{GOT}, given access to $\mathcal{F}_{VOT}$ and $\mathcal{F}_{COM}$, securely realizes $\mathcal{F}_{GOT}$.*

Proof. **(Sender simulation)** The simulator awaits the commands $(\texttt{Send}, k_i, s_i, sid_i)$ from the environment. The simulator then forwards the message $(\texttt{committed})$ to the environment. The simulator awaits for the environment to forward all the $(\texttt{open}, 1, sid_i)$. The simulator checks if shares are consistent, if not, the simulator aborts the protocol. The simulator extracts S and sends $(\texttt{reveal}, S)$ to the Environment. After the simulator received all the z_i, he sends $(\texttt{Send}, z_1 \oplus k_1, \ldots, z_n \oplus k_n)$ to $\mathcal{F}_{GOT}$.

The simulator in the ideal setting and the receiver in the real world are both able to check the validity of shares. Therefore the receiver revealing the secret won't leak any information. It is important to note that aside from the reveal of the commitment, no information is given to sender. Since the simulator can extract the messages, he can input the messages to the ideal functionality.

(Receiver simulation) Simulator generates a random secret s, a set of random k_i and shares $s_i = Share(s)$. Simulator awaits command $(\texttt{transfer}, b_i, sid_i)$ from environment. Simulator sets $d_i = k_i$ when $b_i = 0$ and $d_i = s_i$ when $b_i = 1$ and then forwards to the environment $(\texttt{transferred}, d_i, \texttt{sid})$. The simulator records I as the set of i such that $b_i = 0$. The simulator awaits command $(\texttt{Commit}, S)$ from the environment. The simulator then forwards the message $(\texttt{Reveal}, 1, s_i, sid_i)$ to the environment. On reception of $(\texttt{open})$, checks if $S = S'$ and $I \in \mathcal{I}$, if it is not the case abort. The simulator selects random strings r_i. The simulator sends $(\texttt{Choice}, \texttt{I})$ to $\mathcal{F}_{GOT}$ and for all $i \in I$ sends $(m_i \oplus k_i)$ and for all $i \notin I$ sends r_i.

For the sender to break the protocol, he must learn more messages then he should. This could only occur if he extracts less shares then he should and then correctly guesses the secret. This would force the simulator to abort. Fortunately, the probability of correctly guessing the secret is negligible. We thus have what we need.

□

6.10.2 Applications

The GOT protocols can be used to instantiate k-out-of-n OT using a (n-k)-out-of-n secret sharing. Priced Oblivious Transfer can be instantiated using weighted secret sharing. The price of an object m_i is the weight of the share s_i. Another more complex application is multivariate oblivious polynomial evaluation presented in [Tas11]. Although the GOT protocol in that paper is different, the same techniques in this paper can be used for that

protocol.

6.10.3 Security flaws in previously published protocols based on secret sharing

The protocols from [SSR08] and [Tas11] are insecure against a malicious sender. A malicious sender can easily break the privacy of both schemes. In their protocols, a simple OT is used and share validation is non-existent. As a result, it is possible for the sender to corrupt shares and learn some information about receiver's input from the secret that he reconstructs.

6.11 Batch Single-Choice Cut-and-Choose OT

The Batch Single-Choice Cut-and-Choose OT ($\mathcal{F}_{cacot}$) is an instentiation of $\mathcal{F}_{GOT}$ for a specific enclosed collection. The procedure was introduce in [LP11] and it was used to implement constant round secure function evaluation.

In a $\mathcal{F}_{CACOT}$, the data that will be transferred has a three dimensional structure; a table of pairs. The receiver can learn two category of element of the table. First he can learn exactly all the pair in half the columns. In addition to that, independently for each line, he can either learn the first element of every pair or the second element of every pair. The formal definition of the enclosed collection follows.

Cut-and-Choose OT $\mathcal{F}_{CACOT}$

Input:

— Sender: m_{ijk} where $i \in [1,n], j \in [1,s], k \in \{0,1\}$

— Receiver: $y \in \{0,1\}^n$, $a \in \{0,1\}^{s/2}$

Output:

— Sender: no output

— Receiver: $\{\ m_{ijk} \mid k = y_i\ \} \cup \{\ m_{ijk} \mid j \equiv a_{\lceil j/2 \rceil}\ mod\ 2\}$

Theorem 55 *Any $\mathcal{F}_{CACOT}$ can be implemented with $2ns$ calls to $\mathcal{F}_{VOT}$ where the elements transferred by $\mathcal{F}_{VOT}$ are the maximum between twice the size of the secret and the value of the messages transferred.*

We will now show that there exists an efficient secret sharing scheme with an access structure $\mathcal{C}^*$ such that its complement is the enclosed collection $\mathcal{C}$. We will use a combination

of linear secret sharing (mod p) to share $s = \sigma\text{-}S + J\text{-}S \mod p$, the sum of two secret. Thus, to reconstruct s, the participant will need to reconstruct both σ-S and J-S. Let s_{ijk},where $i \in [n], j \in [s], k \in \{0,1\}$, be the shares of the secret sharing implementing the acces structure $\mathcal{C}^*$. We construct s_{ijk} by the concatenation of σ-S_{ijk} and J-S_{ij} as defined below.

σ-Share: The sender selects a random σ-S. He shares σ-S using a n-out-of-n Secret Sharing resulting in shares σ-S_i and then takes each S_i and distributes each of them twice using an $s/2$-out-of-s secret sharing resulting in shares σ-S_{ij0}, σ-S_{ij1}.

J-Share The sender selects a random J-S and shares it using a $s/2$-out-of-s secret sharing resulting in shares J-S_j. He then shares each J-S_j using a n-out-of-n secret sharing resulting in shares J-S_{ij} where J-S_{ij} is a share of J-S_j.

6.12 Modified Cut-and-Choose from [Lin13]

The Cut-and-Choose OT from [Lin13] is very similar to the one in [LP11]. There are two important differences. First, the set of indices in J is no longer size restricted (instead of size $s/2$). In addition, for each $j \notin J$, the receiver receives a special string v_j which will allow the receiver to prove that $j \notin J$. Although, we could still use the protocol for generalized oblivious transfer defined above, the complement access structure is very complicated. Instead, we will provide a hybrid of the protocols from [Tas11] and [SSR08] to realize this functionality. The protocol follows the same basic approach of the previous protocol: a sharing of a secret, a verifiable oblivious transfer phase, a commitment to a secret, a proof of share validity and finally the message encryption step.

6.12.1 Construction

Essentially, by reconstructing the secret from the sharing that follows the prover will be able to prove two statements. First, it will show that, for each column, the receiver either didn't learn the verification string or one element from each pair. Secondly, it demonstrates that for each row, the receiver either learned the first element of each pair or the second element of each pair. The first statement which can be thought of as a proof of ignorance reflects the approach of [SSR08], while the second one, which can be thought as a proof of knowledge, reflects the approach of [Tas11]. The protocol that follows is thus a hybrid of [Tas11] and [SSR08].

Since the protocol is very similar to the the generalized OT protocol, we will simply describe: how shares are constructed and what is transferred by the Oblivious Transfer.

6.12.2 Sharing

- $s = \mathrm{sr} + \mathrm{sc}$
- $\{\mathrm{sr}_i\} = \mathrm{share}(\mathrm{sr})$ using a n-out-of-n secret sharing scheme
- $\{\mathrm{sr}_{ijk}\} = \mathrm{share}(\mathrm{sr}_i)$ using a s-out-of-s secret sharing scheme twice
- $\{\mathrm{sc}_j\} = \mathrm{share}(\mathrm{sc})$ using a s-out-of-s secret sharing scheme
- $\{\mathrm{sc}_{ij}\} = \mathrm{share}(\mathrm{sc}_j)$ using a n-out-of-n secret sharing scheme
- $(\mathrm{sc}^0_{ijk}, \mathrm{sc}^1_{ijk}) = \mathrm{share}(\mathrm{sc}_{ij})$ using a 2-out-of-2 secret sharing scheme twice
  (so that the receiver can't learn k_{ijb} and $\mathrm{sr}_{ij(1-b)}$ and still be able to reconstruct the secret)

6.12.3 Sender's input to VOT

- $(\texttt{Send}, v_j, \mathrm{sc}_j)$
- $(\texttt{Send}, k_{ijk}, \mathrm{sc}^0_{ijk})$
- $(\texttt{Send}, sr_{ijk}, \mathrm{sc}^1_{ijk})$

6.12.4 Receiver's input to VOT

- if $j \in J$, learn sc_j, sr_{ijk} and k_{ijk}.
- Otherwise if $j \notin J$, learn v_j, $k_{ij\sigma_i}$, $\mathrm{sr}_{ij\sigma_i}$, $\mathrm{sc}^0_{ij(1-\sigma_i)}$ and $\mathrm{sc}^1_{ij(1-\sigma_i)}$.

6.13 Multi-sender k-Out-of-n OT

The Multi-sender k-out-of-n OT functionality was defined in [LOP11] where it was used to optimize the IPS compiler. The functionality involves p sender and one receiver. It is essentially many k-out-of-n OT executed in parallel with the same choice made by the receiver in each execution. This OT primitive can be implemented using ideas similar to the ones used to implement GOT in conjunction with the appropriate use of linear secret sharing.

The protocol is divided in four phases. In the first phases, the senders will construct/distribute the shares of a special secret sharing with value S. They must commit to this information. In the VOT phase, each sender will transfer a key for each message along with the associated share. The receiver will read the key associated with the messages he wishes to learn and otherwise he will obtain a share. The next phase is a verification phase, the sender will commit to S which he could only obtain if he was requesting the same k messages from each sender. The senders will open all their commitment so that the shares are validated by the receiver. If the verification phase succeds, the receiver opens S which proved he only read a legal set of key. In the last phase, the senders will transmit all the messages encrypted with the appropriate key.

The following functionality and protocol involves p senders with n messages of length r each and one receiver. We denote the shares of a a-out-of-b linear secret sharing as $\{B\}_{a\text{-}b}$.

Functionality $\mathcal{F}_{MSOT}$

1. On input $(\texttt{Send}, \langle m_{1j}, \ldots, m_{nj} \rangle)$ from a sender (associated to index j), record all m_{ij}.
2. On input $(\texttt{Transfer}, I \subset [n])$, check if $|I| = k$, if not abort. Send to receiver, for each $j = [p]$ and $i \in I$, the message $(\texttt{Receipt-player}, i, j, m_{ij}))$

Protocol: (π_{MSOT})

Preparation

1. Each sender a selects a random secret S_a and broadcasts a non-interactive commitment to S_a.
   We define $S = \sum_a S_a$.
2. Each sender a reshares S_a to obtain $\{S_{ab}\}_{(n-k)\text{-}n}$.
3. Each sender a reshares each S_{ab} to obtain $\{S_{abc}\}_{p\text{-}p}$.
4. For each j, b and c, sender j sends share S_{jbc} to sender c.
5. Each sender c computes for each b, $S'_{bc} = \sum_a s_{abc}$.
   We have that $S''_b = \{S'_{bc}\}_{p\text{-}p}$ and $\sum S''_b = S$.

VOT's

1. Each sender j selects uniformly at random a set of n one-time pads k_{ij} of length r. He also selects n unused ids denoted by id_{ij} and sends them to the receiver.
2. Each sender j, for each $i \in [n]$ sends $\mathcal{F}_{VOT}$ the command (Send,k_{ij},S'_{ij},id_{ij})
3. Let $I \in \mathcal{I}$ be the set of messages that receiver wishes to receivem he sets $b_i = 0$ if $i \in I$ otherwise he sets $b_i = 1$. For each i, for each sender, receiver sends $\mathcal{F}_{VOT}$ the command $(Transfer, b_i, sid_{ij})$ and records the result.

Verification

1. Receiver computes $S''_b = \{S'_{bc}\}_{p\text{-}p}$ then $S = \sum S''_b$ and broadcasts a non-interactive commitment to S. The receiver commits to a random S if he cannot reconstruct S.
2. Each sender j, for each i, player j opens the commitment to S_{ij}.
3. Receiver verifies that each opened values is consistent with a legal preparation phase and aborts otherwise.
4. Receiver reveals S and if the secret is invalid, the senders abort the protocol.

Transfer

1. Each sender , sends $m_{ij} \oplus k_{ij}$ to receiver who can now calculate m_{ij} for all $i \in I$.

Theorem 56 *π_{MSOT} securely realizes $\mathcal{F}_{MSOT}$.*

6.14 Security of π_{MSOT}

Proof. **(Corrupt senders)** The senders are controlled by the environment. It can be shown that the situation is analagous to a single corrupt sender in the GOT simulator. As such, the protocol for multi-sender k-out-of-n OT against corrupt senders follows the same simulation as a single corrupt sender in GOT. Therefore, the real model with corrupt sender can be simulated using the ideal functionality.

(Corrupt receiver) If the senders are controled by many trusted third parties, this would be undistinguishable from a single trusted third party that controls all the honest parties. It can be shown that for the receiver the situation is analagous to the GOT, As such the real model with corrupt sender can be perferctly simulated using the ideal functionality.

(Corrupt senders and corrupt receiver) The simulator selects random-shares for each honest participant and forwards any sharing to environment. After receiving shares from the

environment, the simulator simulates commitment to the secrets for all honest player. The simulator selects random keys and simulates a $\mathcal{F}_{VOT}$ for each honest party. The simulator awaits that the environment (acting as the receiver) sends a commitment command for a secret S . The simulator reveals the share of honest parties to the environment in a fashion that simulates the open command of VOT. The simulator reveals the commited shares and awaits the open command from the corrupt senders. The simulator awaits the open command from receiver and checks if secret matches the opened shares otherwise abort. The simulator extracts the set of messages that corrupt receiver chose by looking at honest sender's VOT and seing what value the receiver chose from them. The simulator sends F_{MSOT} the receiver's choice and forwards the appropriate messages to the environment. □

6.15 Conclusion

In this paper, we present the first generic constructions of Multi Sender k-out-of-n OT and Batch Single Choice Cut-and-Choose OT. These constructions are based on Generalized OT, which we show how to build from Verifiable OT and proper access structures. Moreover, we formalize structure preserving OT, instantiating a practical protocol and showing how to use it to obtain VOT. This sequence of results provides a novel view the MPC frameworks of [LP11, LP12, LOP11, Lin13], shedding light on the general relations between the primitives used as their main building blocks. As future works we suggest the construction of more efficient general constructions and further investigation of the relations between different flavors of oblivious transfer. An interesting problem in this realm is analyzing the trade-off between share size and number of oblivious transfers in GOT protocols. Moreover, we suggest obtaining versions of our protocols secure against adaptive adversaries building on the recent results of [CKWZ13].

CHAPTER 7

EFFICIENT SECURE TWO-PARTY COMPUTATION FROM BLACK-BOX PRIMITIVES VIA FAULTY CUT-AND-CHOOSE OT

Authors: Samuel Ranellucci, Alain Tapp

7.1 Abstract

We construct a protocol for constant round Two-Party Secure Function Evaluation which improves previous protocols. Our protocol is secure in the standard model, it does not require a random oracle. Our protocol, with the aid of garbled circuits, only requires black-box calls to OT and Commitment. This is achieved by the use of a faulty instantiation of the Cut-and-Choose Oblivious Transfer. The concepts of Garbling Schemes, Oblivious Transfer and Privacy Amplification are combined using the Cut-and-Choose paradigm to obtain the final protocol.

7.2 Introduction

The field of secure multiparty computation studies how to simulate a trusted third party when several players need to compute a function on their private data. Auctions, electronic voting and privacy preserving data analysis are important applications of secure multiparty computation.

The tools that we use in this paper are Oblivious Transfer [Rab81, EGL85], Commitment Schemes [Blu83, Eve83] and some variation of Yao's Garbled Circuit [Yao82, BHR12]. In this work, we consider malicious adversaries and a strong notion of security, called Universal Composability [Can01]. The goal of this work is to present a very efficient protocol for constant round Two-Party Secure Function Evaluation in this model based on the above primitives.

Important work has gone into optimizing the primitives of secure computation. Oblivious Transfer requires expensive public key operations to instantiate from scratch, thus the idea of generating a large number of them from a much smaller number of OT by using private key operations is extremely valuable. This concept is called OT extension [Bea96, IKNP03]

and the fact that the protocol we present in this chapter is based on black-box OT insures that these types of optimization apply.

Garbling schemes [Yao82] are convenient and efficient constructions and have been the basis of many interesting two-party computation protocols [FJN+13, KSS12, LP11, SS11, LP07] but other approaches have been used [IPS08, LOP11, NO09, NNOB12].

7.2.0.1 Our results in perspective

In recent times, Secure Function Evaluation with malicious adversaries using Garbled Circuits have become more and more practical. Previous protocols based on garbling schemes following the Cut-and-Choose paradigm required $\Omega(ns)$ calls to OT. Lindell and Pinkas started the trend with their work in [LP07] that required $\Omega(ns)$ OT but $\Omega(ns^2)$ Commitments. The work of [LP11] and [SS11] only require $\Omega(ns)$ commitments but require stronger notions of OT such as committing OT and Batch Single Choice Cut and Choose OT. These protocols do not admit OT extension. The results of [FJN+13] allows OT extension and requires $\Omega(cs/log(c))$ commitments but reduces the number of garbled gates by a $\log(c)$ factor. The result [Fre12], [MR13] and [KSS12] allow OT extensions but the first two are in the Random Oracle model and the last one relies on Claw-Free Collections. In contrast, our work only uses $O(n+s)$ OT, allows OT extension and does not rely on additional assumptions. Note that OT extension does not provide OT for free.

paper	OT	Commit	OT extension	Assumptions
[LP07]	ns	ns^2	yes	None
[LP11]	ns	ns	no	DDH
[SS11]	ns	ns	no	Claw-Free Collection, DDH
[Fre12, FN13]	ns	ns	yes	Random oracle
[KSS12]	ns	ns	yes	Claw-Free Collection
[FJN+13]	$n+s$	$cs/log(c)*$	yes	Random oracle
[MR13]	ns	ns	yes	Random oracle
[Lin13]	ns	ns	no	DDH
[HKE13a]	ns	ns	no	DDH, Random oracle
this work	$n+s$	ns	yes	None

7.2.0.2 Notation and convention

We will denote the adversary's view of a player's input as a random variable. We will refer to 1-out-of-2 Oblivious String Transfer for string of length l as OT. The security parameter will be denoted by s. The number of gates of the circuit will be denoted by c and its input size by n. Uniformly random and independent choices for x in a set X is denoted $x \in_R X$. The notation $x \in_R A$ describes the random selection of an element x according to random Variable A. The range of an integer starting from i and ending with j is denoted $[i, j]$. As usual, we define secure function evaluation $\mathcal{F}_{SFE}$ as a two-party computation where Bob learns the output.

7.2.0.3 Structure of the paper

We first present our notation for the **Garbling schemes**. This leads to a very good generic protocol for **two-party computation** using perfect Cut-and-Chose OT. We then present a **Faulty Cut-and-Choose OT**. The drawback of using this faulty primitive is formalized in the **Selective failure attack** section. This is followed by our **Main protocol** where we propose a very efficient and secure protocol based on the Faulty Cut-and-Choose OT. The security proof in the universal composability paradigm follows (**sender simulation** and **receiver simulation**).

7.3 Garbling schemes

In 1982, Yao generated a construction that came to be known as Yao's garbled circuit. The idea was to encode a circuit in such a way that by giving a party the right keys he could evaluate the circuit on a specific input and yet learn nothing about the input except what could be deduced from the output. This construction came to be used in many different applications each with their own divergent variant. Lindell and Pinkas were the first to prove the security of Yao's garbled circuit in the field of two-party computation [LP09]. Garbling schemes proposed in [BHR12] define the notions that unify these different variants.

We will need to distinguish between a function $f : \{0,1\}^n \to \{0,1\}^m$ and its representation $\bar{f}$ (in our case we can restrain ourselves to circuits). A garbling algorithm GB is a randomized algorithm that transforms $\bar{f}$ into a triple of functions $GB(\bar{f}) = (F, e, d)$. We require that $f = d.F.e$. The encoding function e turns an initial input $x \in \{0,1\}^n$ into a garbled input

$X = e(x)$. Evaluating the garbled function F on the garbled input X gives a garbled output $Y = F(X)$. The decoding function d turns Y into the final output $y = d(Y)$, which must coincide with $f(x)$. One has probabilistically factored f into $d.F.e$. Thus a garbling scheme $G = (GB, En, De, Ev, ev)$ is regarded as a five-tuple of algorithms, with strings $\bar{d}$, $\bar{e}$, $\bar{f}$, and $\bar{F}$ interpreted as functions under the auspices of functions De, En, ev, and Ev.

A circuit garbling scheme is a five-tuple of algorithms $\mathcal{G} = (GB, En, De, Ev, ev)$. The first two algorithms are probabilistic, the remaining are deterministic. The string $\bar{f}$ represents the function f that we wish to garble. On input $\bar{f}$, the algorithm GB outputs an encoding function $En(\bar{e}, .)$ which maps an initial input $x \in \{0,1\}^n$ to a garbled input $X = En(\bar{e}, x)$, as well as $\bar{F}$ that describes a garbled function which maps each garbled input X into a garbled output Y, and a string $\bar{d}$ describing a decoding function $DE(\bar{d}, .)$ which maps each garbled output Y into a a final output $y = De(\bar{d}, Y)$.

The side-information function is an important feature of garbling schemes. In a protocol that follows, Alice will have to garble a circuit. Bob will have to verify that part of the output is independent from part of his input. The only way this can be achieved is by looking at the underlying side-information.

Definition 5 *A garbling scheme is* **correct** *if for any function f the application of GB results in strings $(\bar{\mathcal{F}}, \bar{e}, \bar{d})$ such that $De(\bar{d}, Ev(\bar{\mathcal{F}}, En(\bar{e}, x))) = ev(\bar{f}, x)$.*

Definition 6 *A garbling scheme with an encoding function $En(\bar{e}, .)$ with input $x = x_1 \dots x_n$ where $x_i \in \{0,1\}$ is* **projective** *if for every $\bar{e}$ there exists a list of tokens $(T_1^0, T_1^1, \dots, T_n^0, T_n^1)$ such that $En(\bar{e}, x) = (T_1^{x_1}, \dots, T_n^{x_n})$.*

Definition 7 *A projective scheme is* **transparent**, *if given $\bar{f}$, $\bar{\mathcal{F}}$,and the token $(T_1^0, T_1^1, \dots, T_n^0, T_n^1)$ in a random order for each variable, one can efficiently verify that $\bar{\mathcal{F}}$ is consistent with $GB(\bar{f})$ and furthermore the order of the tokens can be deduced.*

Definition 8 *The side-information function ψ denotes the information revealed about $\bar{f}$ from $(\bar{F}, \bar{e}, \bar{d})$.*

Definition 9 *A garbling scheme is private relative to a topological side information function ψ if for any inputs x_0,x_1 and description of functions $\bar{f}_0, \bar{f}_1$ such that $ev(\bar{f}_0, x_0) = ev(\bar{f}_1, x_1)$ and $\psi(\bar{f}_0) = \psi(\bar{f}_1)$, the adversary cannot distinguish between $(\bar{F}_0, \bar{e}_0, \bar{d}_0, X_0)$ generated from $(\bar{f}_0, x_0)$ and $(\bar{F}_1, \bar{e}_1, \bar{d}_1, \bar{X}_1)$ from $(\bar{f}_1, x_1)$.*

Definition 10 *A garbling scheme is malleable if for any function* $\bar{f}$ *such that* $GB(\bar{f}) = (\bar{F}, \bar{e}, \bar{d})$ *and every* $y \in \{0,1\}^m$ *there exists a* $\bar{g}$ *such that* $\forall x, g(x) = y$ *and* $\psi(\bar{f}) = \psi(\bar{g})$.

7.4 Two-party computation

The first application of a Garbling scheme was presented by Yao in order to implement secure two-party computation. In a context where the players are semi-honest, by using OT it becomes easy for Alice and Bob to evaluate a circuit of their choice securely on their private inputs. Alice can construct the garbled circuit and send it to Bob with the keys associated to her input. Alice then transmits to Bob his keys using OT. For each of Bob's input bits, Alice will send the key associated with the value 0 and the key associated with the value 1. For each of his input bits, Bob will choose to learn the key corresponding to his input at that position. Since Bob has all the input keys as well as the circuit, he can now perform the evaluation and extract the result.

In order to make this scheme secure against an active adversary, several issues have to be addressed. In terms of correctness, Bob must know that the circuit he evaluates computes the right function. This is not sufficient since Alice could transfer keys that are correct only for certain input bit and thus learn information about Bob's input following the success or failure of the protocol. Bob must know that the protocol would have worked correctly even if he had chosen another input. These two requirements can be ensured by using what is called Batch-Single Choice Cut-and-Chose OT introduced in [LP11].

Informally, Alice will create s circuits with all of their associated input keys. Half of the circuits will be completely revealed so that Bob can verify that they are correct. It is thus unlikely that this phase succeeds and that more than a few of the remaining unopened circuit are incorrect. By evaluating the remaining circuits and taking the most common result as the output, correctness is guaranteed. To keep Alice's input private, it is important that Bob always evaluates the circuit with the same input. Combining all these requirements, we see that the generic Oblivious Transfer has to be replaced with something with more structure. The set of pairs of keys for all the circuits will be organized in a table. Each of the s columns is associated with a circuit. Each row is associated with an input variable (of Bob) and each element of the table is a pair of keys, which are respectively associated with the values 0 and 1. Learning half of the circuits (check circuits) means Bob must learn for exactly half of the

columns the value of both keys. The fact that Bob always uses the same inputs bits means that for every individual row he must choose between learning the first element of each pair or learning the second element of each pair. In the following section, we will present a variant of that primitive that we call Cut-and-Chose OT.

Bob can now reveal the identity of the check circuit so that Alice sends him all the associated input keys. With all the keys he can verify the correctness of those circuits. The rest of the circuits are going to be evaluation circuits and for all of those, Alice will send the keys associated to her input. By choosing different inputs for some evaluation circuit, Alice can learn information about the sensitivity of the circuit conditioned on Bob's input. To solve this problem [Fre12, FN13] introduced a verification functions v. Instead of just generating garbled circuits for f, garbled circuits for $f'[(x,a),(y,b)] = (f[x,y], v[x,a,b])$ will be generated. The following choice of functions has been introduced.

Definition 11 *We will denote $z_i(x,a,b) = (p^i(x) \cdot b) \oplus a_i$ and $v(x,a,b) = (z_0, \ldots, z_n)$ as the verification function. Where p^i is the cyclic shift toward the right of i position and $\cdot$ the inner product mod 2.*

It was proven in [Fre12] that v does not reveal information about Alice's inputs and at the same time guarantees that inconsistent inputs will be detected except with exponentially small probability.

7.5 Faulty Cut-and-Choose OT

In this section, we will introduce the Cut-and-Choose OT which is a faulty variant of the Batch-Single Choice Cut-and-Chose OT introduced by [LP11]. This is a variant of OT where more than 2 messages are sent simultaneously and where the choice of messages received by Bob has a special structure. The input from the sender is a table M of ns pairs of binary messages (m_{ij0}, m_{ij1}) of length l. Using our faulty protocol, an honest sender would learn nothing and an honest receiver would learn exactly the following. For each row, the receiver selects a y_i and gets $\{m_{ijk} \mid k = y_i\}$. For each odd index of column j, the sender selects an $a_j \in \{0,1\}$ and learns all messages in $\{m_{i(j+a_j)k}\}$.

We implement a faulty variant of this primitive from $O(n+s)$ OT. The transfer of rows and the transfer of columns will be done independently and thus an honest Alice has to use the same information twice. Each of the n rows will incur one OT where the first string will

be the first element of each pair in that row and the second string will be the second element of each pair. For each pair of columns, each column will be contained in a single string. The sender and the receiver will execute s OT with the strings associated to the columns. If the sender is honest, the receiver will get exactly what he requested. One of the problems of the protocol is that the choice of Bob can be adaptive. To prevent this, Alice will send keys (One time pads) instead of messages. Once every key is transmitted, she will send the messages encrypted with different keys. Another problem with this protocol is that the sender can choose the elements of the row inconsistent with the columns. The receiver, if he finds an inconsistency, will have to abort but this can be done at a later time. Instead, for the time being, he will select the column value as the output. This is essential to the main protocol. In any case, the success or failure of the verification step leaks information about Bob's choice and if used in the protocol described previously, it would leak information about Bob's input. This issue will be dealt with in the final protocol. The following collection denotes the sets of messages that the receiver can pick and where he learns exactly all the messages in the chosen set.

Definition 12 *Each messages is denoted as* m_{ijk} *where* $i \in [1,n], j \in [1,s], k \in \{0,1\}$.

- $\mathcal{I} = I_0 \cup I_1$
- $I_0 = \{ \{m_{ijk} \mid k = y_i \} \mid y_i \in \{0,1\} \}$
- $I_1 = \{ \{m_{i(j+a_j)k} \mid a_j \in \{0,1\} \} \mid j \in [1,3,\ldots,s-3,s-1]\}$
- $p_{ic} = p_{i1c}, ..., p_{isc}$
- $w_j = p_{1j0}, p_{1j1}, ...p_{nj0}, p_{nj1}$

The following protocol implements the Cut-and-Choose OT.

Protocol: π_{CCOT}

1. For each $i \in [1,n], j \in [1,s], k \in \{0,1\}$, the sender selects $p_{ijc} \in_R \{0,1\}^l$.
2. For each $i \in [1,n]$, the sender and the receiver execute $\mathcal{F}_{OT}$ with messages (p_{i0}, p_{i1}) with choice bit y_i.
3. For each odd $j \in [1,s]$, the sender and the receiver execute $\mathcal{F}_{OT}$ with messages (w_j, w_{j+1}) and choice bit a_j.
4. The receiver checks that the values he received (the p_{ic} and $w_{j'}$) are consistent. If not, he notes the inconsistency and uses the column value as the output (the $w_{j'}$).

5. The sender sends $z_{ijk} = m_{ijk} \oplus p_{ijk}$.

The protocol is faulty and therefore its security will not be proven independently in this section. That being said, the problem only arises in the case of a cheating sender and therefore we can still discuss the possibility of simulating the receiver.

Theorem 57 *A simulator having access to the ideal functionality for Cut-And-Chose OT, when the environment controls the receiver, can simulate the view of the environment in the real world.*

Proof. The simulator selects $p_{ijc} \in_R \{0,1\}^l$. The simulator awaits the environment's choice (y_i and a_j) for each OT. (The simulator is able to extract the choice bits in the OTs for each row and column pair.) The simulator forwards the appropriate combination of keys as specified in the protocol. The simulator calls the ideal functionality with these choices. For each message m_{ijk} that the simulator has obtained from the ideal functionality, he uses the key p_{ijk} to encrypt those messages and sends the cyphertext $z_{ijk} = m_{ijk} \oplus p_{ijk}$ to the environment. For the messages m_{ijk} that the simulator has not extracted, he forwards a random message z_{ijk}.

The distribution of m_{ijk}, z_{ijk}, p_{ijk} in the simulation is identical to the distribution of the real world. □

7.6 Selective Failure Attack

In order to achieve a very high level of efficiency in our final protocol, we will have to combine several ideas. One of them is to use a faulty Cut-And-Chose OT. In our protocol, Alice can gain information about Bob's input by misbehaving and waiting to see if this causes the protocol to abort.

In general, a selective failure is a type of attack where the adversary makes the successful completion of the protocol depend on the other player's input. In our case, by making sure that the selective failure attack only leaks one bit (whether or not the protocol failed), we can later use privacy amplification to render this information useless to Alice.

The following definitions and lemmas will be used in the security proof of our protocol ([DKRS06, DORS08]). Let et A, B be random variables and U_ℓ the uniform distribution over $\{0,1\}^\ell$.

Definition 13 $H_\infty(A) := -\log \max_{a \in A} (\Pr[A = a])$

Definition 14 $\bar{H}_\infty(A|B) := -\log \underset{b \in B}{E} \max_{a \in A} \Pr[A = a \mid B = b]$

Lemma 58 *If B has at most 2^λ values, then $\bar{H}_\infty(A|B) \geq H_\infty(A) - \lambda$.*

Lemma 59 *Assume H is a class of universal hash functions from $\{0,1\}^n$ to $\{0,1\}^\ell$, for any random variable A, B such that $\bar{H}_\infty(A|B) \geq k$. We have that for a randomly selected $h \in \mathcal{H}$, the distributions of $(h(A), \mathcal{H}, \mathcal{B})$ and $(U_\ell, \mathcal{H}, \mathcal{B})$ are ϵ-close where $\epsilon = 2^{-\frac{1}{2}(k-l)+1}$.*

7.7 Main protocol

Although the complete protocol requires many steps, the right combination of ingredients results in a very efficient protocol based on general assumptions. In the previous section, we explained a known approach to combine Cut-and-Choose OT and Garbling Schemes to obtain Two-Party Secure Computation. Our general protocol will follow this approach. Without further modification of the protocol, the privacy of the receiver's input would be jeopardized by the faultiness of the Cut-and-Chose OT. We will thus modify the general protocol to address this problem. The function to be evaluated will be modified in such a way that the leakage contains no information on Bob's input.

The function of two players, which is to be evaluated, will be replaced by $f'(x, y', h, q) = f(x, h(y') \oplus q) = f(x, y)$. To extract the output, y' will be chosen uniformly at random, h will be taken randomly from a family of universal hash functions with the appropriate parameters and $q = h(y') \oplus y$. Since h and q do not contain information about the input, they can (and will) be declared publicly after the faulty Cut-and-Chose OT has been used to transfer y' among all the different circuits. Thus, Alice's input will remain the same, but Bob's input is now decomposed into (y', h, q), where only y' is transferred using the Cut-and-Choose OT. Intuitively, since only a limited amount of information about y' is leaked, privacy amplification ensures that y remains private.

Another important technicality has to be taken care of. We require that the topological leakage function of the garbling scheme show that for each garbled circuit that the output of the verification function is independent of (h, q, y'). In other words, that the tokens for (h, q, y') do not affect the output of the validation function. If this was not verified, then a corrupted sender could simply make the consistency of the output for the verification function

depend on y. This would allow the sender to learn information based on a selective failure attack.

Definition 15 *We denote $\mathcal{H}$ a family of Universal Hash Function from $\{0,1\}^{n+2s+1}$ to $\{0,1\}^n$.*

Any universal Hash Function Family can be used in our protocol but the one presented in [Tho09] is compact and efficient.

Definition 16 *Denote $f'((x,a),(y',h,q,b)) = (f(x,h(y')\oplus q), v(x,a,b))$ where h is in $\mathcal{H}$ and v is the verification function that was defined in the previous section.*

In our main protocol, any circuit garbling scheme that is correct, private, malleable, transparent, and projective relative to a topological side information function ψ can be used. Garbling schemes similar to Yao's garbled circuit normally meet those requirements. For explicit constructions, see [LP07, BHR12]. We will also use the following length parameters, the length of the input to the function to be evaluated is denoted as usual by n. We will denote $n_q = n$, $n_h = 2(n+2s+1)$, $n_x = n$, $n_a = n$, $n_y = 2n+2s+1$ and $n_t = n_q+n_x+n_y+n_h$. To clarify the role of each player, we will refer to Alice as the sender and Bob as the receiver.

Protocol: π_{SFE}

1. The sender extracts s triplets $GB(f') \rightarrow (F_j, e_j, d_j)$.
2. For each circuit j, using the encoding functions e_j, the sender extracts all of the tokens associated to both values for all inputs. Tokens associated to the sender will be denoted by $(x^0_{1j}, x^1_{1j}, \ldots, x^0_{n_xj}, x^1_{n_xj})$, $(a^0_{1j}, a^1_{1j}, \ldots, a^0_{n_aj}, a^1_{n_aj})$, those to y' by $(y^0_{1j}, y^1_{1j}, \ldots, y^0_{n_yj}, y^1_{n_yj})$, those to the hash function by $(h^0_{1j}, h^1_{1j}, \ldots, h^0_{n_hj}, h^1_{n_hj})$, those associated to q by $(q^0_{1j}, q^1_{1j}, \ldots, q^0_{n_qj}, q^1_{n_qj})$ and those to b by $(b^0_{1j}, b^1_{1j}, \ldots, b^0_{n_bj}, b^1_{n_bj})$.
3. The receiver selects $y' \in_R \{0,1\}^{n+2s+1}$, $q \in_R \{0,1\}^n$, $h \in_R \mathcal{H}$ and $J \in_R \{0,1\}^{s/2}$. The receiver sets $q = h(y') \oplus y$ and commits to h and q. (J represent the check circuit selection)
4. The sender and the receiver execute π_{CCOT} where the sender's input is $(y^0_{1j}, y^1_{1j}, \ldots, y^0_{n_yj}, y^1_{n_yj})$ and the receiver's input is the set consistent with y' and J. (as mentioned earlier, in case of inconsistency, the receiver chooses the column value)
5. For each circuit $j \in [1,s]$ the sender performs the following steps

— send F_j, d_j to the receiver

— for each $i \in [1, n_h]$, commit to h_{ij}^0 and h_{ij}^1

— for each $i \in [1, n_b]$, commit to b_{ij}^0 and b_{ij}^1

— for each $i \in [1, n]$, select $r_{ij} \in_R \{0, 1\}$ and commit to $(X_{ij}^0, X_{ij}^1) = (x_{ij}^{r_{ij}}, x_{ij}^{1 \oplus r_{ij}})$

— for each $i \in [1, n_a]$, select $r'_{ij} \in_R \{0, 1\}$ and commit to $(A_{ij}^0, A_{ij}^1) = (a_{ij}^{r'_{ij}}, a_{ij}^{1 \oplus r'_{ij}})$

6. For each $j \in [1, s]$, the receiver checks from F_j, d_j and ψ that the output of the verification function is not affected by the tokens for (h, q, y'). (ψ is the topological side information function)
7. The receiver reveals J to the sender. For each $j \in J$ and each input $i \in [1, n]$, he sends y_{ij}^0, y_{ij}^1 to the sender.
8. The sender aborts if the values he received in the previous step differ from what he had sent.
9. For all $j \in J$, and i having the appropriate range, the sender opens the commitments $(X_{ij}^0, X_{ij}^1), (A_{ij}^0, A_{ij}^1), (h_{ij}^0, h_{ij}^1), (q_{ij}^0, q_{ij}^1), (b_{ij}^0, b_{ij}^1)$.
10. For all $j \in J$, the receiver checks that the garbled circuit j is valid. The receiver aborts at this point if a garbled circuit is faulty or if he had noticed any inconsistency during the cut-and-choose protocol.
11. For all $j \notin J$, and all $i \in [1, n]$, the sender opens $X_{ij}^{x_j \oplus r_{ij}} = x_{ij}^{x_j}$. For all $j \notin J$, and all $i \in [1, s]$, the sender opens $X_{ij}^{x_j \oplus r_{ij}} = x_{ij}^{x_j}$ and $A_{ij}^{a_j \oplus r'_{ij}} = a_{ij}^{a_j}$.
12. The receiver opens his commitments to h and q and declares a value b.
13. The sender opens the commitments to the token associated to the values h, q, b.
14. The receiver uses the tokens he received as well as the pairs (F_j, d_j) to do the evaluations using the garbling scheme. He checks if the validation outputs are consistent and aborts if they are not. The receiver takes the most common value as his output.

7.8 Sender simulation

The sender does not receive any output and the receiver mostly reveals randomly chosen values in the protocol. Therefore the simulator job consists mostly of extracting the sender's input, aborting with the same distribution as in the real world and revealing random values.

— The simulator sends the messages (`committed`) which are associated to the receiver committing to h and q. (3)

— The simulator selects uniformly at random the set J (of size $s/2$) and the string y' and derives an associated set I.

— The simulator simulates an execution of π_{CCOT} and extracts the environment's input to π_{CCOT} . (4)

— The simulator awaits the commitment command with the keys associated to the circuits as well as each (F_j, d_j). (5)

— The simulator forwards J as well as all the keys whose indices are matched to J. (7)

— The simulator awaits that the environment sends the open command associated to all the keys in the check circuit. The simulator aborts using the same criteria as an honest receiver. He aborts if any of the check circuits is incorrect or if the outputs of the OTs in the π_{CCOT} are inconsistent.

— The simulator awaits that the environment sends the open command to the keys associated to his choice of input. (11)

— The simulator chooses uniformly at random h, q and b and forwards them to the environment (h,q as a reveal command from $\mathcal{F}_{COM}$). (12)

— The simulator awaits that the environment sends the open commands for Commitments to keys associated to the given h, q and b. (13)

— The simulator evaluates the evaluation circuits and checks that all validation outputs ($v(.)$) are consistent; if not he aborts.

— The simulator has all the tokens, because of the transparency of the garbling scheme, he can determine for each circuit and each input the value of input associated to each key. For each of the sender's input bit, he takes as input bit the value that appears in the majority of evaluation circuits. He can thus extract the input and send it to the ideal functionality. (14)

Lemma 60 *If we denote Y' as a random variable denoting the honest sender's choice for y' (input to Faulty Cut-and-Choose OT). After step 10, the environment's view of $h(Y')$ in the real world is 2^{-s}-close to the uniform distribution over $\{0,1\}^n$.*

Proof. First, we note that until step 10, the environment's view is independent of Y'. In step 10, we note that the environment already knows if the protocol would have failed because of a

faulty check circuit. If it is the case, then the protocol would have aborted anyways and $h(Y')$ looks exactly like the uniform distribution. Otherwise, the environment learns whether or not the receiver decided to abort. This is modeled as a variable Z. Since each pair of columns and each row was obliviously transfered, the only information about y' that the environment has is Z. Z has a one bit domain and thus $\bar{H}_\infty(Y'|Z) \geq H_\infty(Y') - 1 = n + 2s + 1$. Therefore by lemma 58, we have what we want. □

7.8.1 Simulation Validity

It is evident to see that if the environment does not corrupt circuits, send inconsistent shares in Faulty cut-and-choose OT or open tokens which results in different inputs for different evaluation circuits, the environment will be unable to distinguish between the real or ideal setting. The distribution of all variables is identical and neither the simulator nor the real world execution will abort.

We first address the case where the environment only corrupted some circuits. This would make the simulation fail if the probability of having the simulation abort is different from the probability in the real world execution or if the simulation does not abort but is unable to extract the relevant information. In the simulation (as in the honest protocol), each circuit is a check circuit with probability one-half. Since the receiver does not reveal any inconsistency in the cut-and-choose OT until after he has declared the choice of check circuit, the sender must corrupt circuits in such a way that more than half of evaluation circuits are corrupted and none of the check circuits are corrupted. Otherwise the simulator aborts as in the real world setting. This can only happen with exponentially small probability.

We now address the case where the sender provides inconsistent inputs to the OT in the Cut-and-Choose OT protocol.

The keys that are returned by the receiver for the check circuits only depend on the environment's choice of values in the columns (see π_{CCOT}). Since an honest receiver always knows those values by virtue of being associated to check circuits. The simulator can perfectly simulate that step by simply revealing what an honest receiver would know in the real world.

We will show that the leakage does not allow the environment to distinguish between the ideal world and the real world. If the simulator notices any inconsistency during the Cut-and-Choose protocol or that a garbled circuit was faulty, he aborts. Since the check circuits are public at this point, the environment knows that either a corrupted circuit was

a check circuit (case we already dealt with) or that the receiver (simulator or honest party) has selected an input such that the row and column are inconsistent. The key factor is that the probability that the protocol aborts depends only on the choice of inconsistency and is the same for the simulation as for the real world.

We can see (by lemma 59) that in the real world $h(y')$ is indistinguishable from a value uniformly selected at random from $\{0,1\}^n$. As such, the simulator cannot distinguish between a q which depends on y (real world) versus one that is generated uniformly at random (ideal world).

Finally, if the environment tries to send inconsistent inputs for two circuits and the circuits are valid, the validation step will result in an abort except with exponentially small probability. This abort, due to the fact that the receiver verified that tokens for (y', h, q) do not affect the output of the verification function, is independent of Bob's input. Thus we can see that sending inconsistent inputs will not allow the environment to distinguish between the real and ideal setting.

7.9 Receiver Simulation

— The simulator awaits the commit command associated to h and q. (3)

— The simulator awaits the receiver's input for π_{CCOT} from the environment, the simulator notes the receiver's choice of check circuit as J as well as the input y' and sets $y = h(y') \oplus q$.

— For each $j \in J$, the simulator sets $(F_j, d_j, e_j) = GB(f')$.

— The simulator selects r uniformly at random. Let $g'[(x,y),(h,y',r)] = (z,r)$. Because of the malleability of the scheme, the simulator can produce g consistent with $GB(g')$ such that $\psi(f') = \psi(g)$. For each $j \notin J$, the sender sets $(F_j, d_j, e_j) = GB(g)$.

— The simulator sends the appropriate tokens that are associated with the environment's input in the π_{CCOT} . He also forwards the Commitment messages to the remaining tokens as well as all the (F_j, d_j, e_j) he has constructed. (5)

— The simulator awaits the environment's choice of check circuits as well as the inputs he received in the π_{CCOT} . The simulator checks if these values are consistent with what he has sent, if it is not the case, the simulator aborts. (7)

— The simulator sends the reveal messages for all of tokens associated to check circuits.

(11)

— The simulator randomly selects for each input and each circuit to send a reveal message for one of the tokens. (12)

— The simulator awaits the open commands to h and q. The simulator simulates the opening of all the tokens required by the protocol. (14)

First we note that in the case of a corrupted receiver, the simulator can simulate the faulty variant of cut-and-choose OT using a cut-and-choose OT functionality. The simulator can trivially extract the environment's input from the choice of input to the faulty variant of cut-and-choose OT, as well as his commitments to h and q. The main difference between the ideal setting and the real setting is the construction of a fake garbled circuit. Fortunately, due to the privacy of garbling schemes, these do not allow the environment to distinguish which model he is part of. The only remaining recourse for the environment is to try to lie about which circuits are check circuits. But since the probably of guessing the keys is negligible, we can see that the real and ideal setting are indistinguishable.

7.10 Circuit Optimization

We believe that the strong point of the two-party computation protocol presented in this chapter is its efficiency. In this section, we will explain why some recent work on optimization of protocols based on garbled circuits can be applied to our protocol. In [Lin13], a very nice optimization technique is presented for achieving secure computation while reducing the total number of circuits that have to be transmitted to s and still achieving security of 2^{-s}. Another approach was presented in [HKE13a] but Lindell's technique applies more naturally to our protocol. The detailed presentation of the technique requires a full length publication and therefore we will assume in this section that the reader is familiar with Lindell's article.

The basic idea of this optimization is as follows: the protocol will be divided in two computations, an **output extraction** and a **cheating sender input extraction**. In the **output extraction** phase, the receiver will be able to extract the output of the functionality or a proof that the sender has cheated. In the **cheating sender input extraction** phase, the sender inputs the same value as in the **output extraction**, if the receiver has acquired a proof of cheating, he is able to extract the sender's input and otherwise he gets nothing. Note that having the receiver obtain the sender's input enables the protocol to be correct

in a trivial way. It is necessary to weave these two computations together so that certain properties are verified. First, the protocol must ensure that the sender uses the same input in both computations. Second, it is also needed that evaluation of the garbled circuits of both computations is done before checking either of them. This is to insure that a receiver can't extract a proof of cheating from an honest sender.

A necessary building block of the construction is the modified cut-and-choose OT similar to the one in [LP11]. There are two important differences. First, the set of indices in J is no longer size restricted (instead of size exactly $s/2$). In addition, for each $j \notin J$, the receiver receives a special string which allows him to prove that $j \notin J$.

We will now describe how the main protocol can be modified to enable this optimization. Since we do not have the space to formally describe the construction, we will instead provide a high level overview of the changes required.

Instead of implementing a modified cut-and-choose directly, our protocol implements a faulty modified cut-and-choose OT. We thus have to adapt the ideas from the main protocol to deal with the faultiness and leakage. We have to tangle the computations, delay the input verification of the **output extraction** and merge it with the input consistency of **cheating sender input extraction**.

7.10.0.1 Faulty Modified Cut-and-Choose OT

The faulty version of modified cut-and-choose OT can easily be implemented by a minor modification of the faulty cut-and-choose OT protocol. Instead of doing one OT per pair of column, one simply does an OT for each column where the first element transferred in the OT is the value of the column and the second element is the secret that allows the sender to proves that $j \notin J$.

7.10.0.2 Sender input consistency

It is not to hard to realize that in the main protocol, the receiver can extract an output (even if it is an error) and then use the consistency check to verify that the outputs were all consistent. It is thus possible to merge the consistency check of **output extraction** and the consistency check of **cheating sender input extraction** together, in order to verify that the sender uses the same input in both computations.

7.10.0.3 Verification of circuits

It is important to note that in our main protocol, once the faulty cut-and-choose has taken place and the sender has committed to his inputs, the circuits are fully determined. As such, we can delay the checking phase after evaluation has taken place.

7.11 Conclusion

Although our protocol is very efficient, we explain why some recent optimization work also apply to our protocol. We believe some further optimizations are possible. The inclusion of the hash function into the function to be computed could increase the size of the circuit, especially small ones. Although it is not straightforward, we believe that by using the appropriate garbling scheme coupled with a family of hash function with a nice circuit structure, one could hardcode the hash function into the circuit. Another possible optimization could be to select a class of universal hash functions which mends well with the free XOR technique [KS08]. The construction of universal hash function from random binary matrices seems to be an ideal choice for these two optimizations. We believe that the idea of using faulty primitives with adapted protocols can improve secure computation also beyond two-party secure function evaluation.

BIBLIOGRAPHY

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *Advances in Cryptology–CRYPTO 2010*, pages 209–236. Springer, 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *Advances in Cryptology–CRYPTO 2011*, pages 649–666. Springer, 2011.

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *Advances in Cryptology–EUROCRYPT 2012*, pages 483–501. Springer, 2012.

[ALR11] Gilad Asharov, Yehuda Lindell, and Tal Rabin. Perfectly-secure multiplication for any $t < n/3$. *Advances in Cryptology–CRYPTO 2011*, pages 240–258, 2011.

[BBCM95] Charles Bennett, Gilles Brassard, Claude Crépeau, and Ueli Maurer. Generalized privacy amplification. *Information Theory, IEEE Transactions on*, 41(6):1915–1923, 1995.

[BBCS92] Charles Bennett, Gilles Brassard, Claude Crépeau, and Marie-Hélène Skubriszewska. Practical quantum oblivious transfer. In *Advances in Cryptology-CRYPTO 91*, pages 351–366. Springer, 1992.

[BBR88] Charles Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM journal on Computing*, 17(2):210–229, 1988.

[BC87] Gilles Brassard and Claude Crépeau. Zero-knowledge simulation of boolean circuits. In *Advances in Cryptology-CRYPTO 86*, pages 223–233. Springer, 1987.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. *Advances in Cryptology-CRYPTO 2009*, pages 108–125, 2009.

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt. Nielsen, Jakob. Pagter, and Toft Tomas Schwartzback, Michael. Secure multiparty computation goes live. *Financial Cryptography and Data Security*, pages 325–343, 2009.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC'08*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable vrfs revisited. *Pairing-Based Cryptography–Pairing 2009*, pages 114–131, 2009.

[BCR87] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *Advances in CryptologyCRYPTO 86*, pages 234–238. Springer, 1987.

[BCS96] Gilles Brassard, Claude Crépeau, and Miklos Santha. Oblivious transfers and intersecting codes. *Information Theory, IEEE Transactions on*, 42(6):1769–1780, 1996.

[BCW03] Gilles Brassard, Claude Crépeau, and Stefan Wolf. Oblivious transfers and privacy amplification. *Journal of Cryptology*, 16(4):219–237, 2003.

[BDNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In *Advances in Cryptology-CRYPT095*, pages 97–109. Springer, 1995.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 479–488. ACM, 1996.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[BHR12] Mihir Bellare, Viet Tung. Hoang, and Philip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796. ACM, 2012.

[BJRT12] Anne Broadbent, Stacey Jeffery, Samuel Ranellucci, and Alain Tapp. Trading robustness for correctness and privacy in certain multiparty computations, beyond an honest majority. In *Information Theoretic Security*, pages 14–36. Springer, 2012.

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[BM90] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 547–557. Springer, 1990.

[BM04] Amos Beimel and Tal Malkin. A quantitative approach to reductions in secure computation. In *Theory of Cryptography*, pages 238–257. Springer, 2004.

[BMSW02] Carlo Blundo, Barbara Masucci, Douglas R. Stinson, and Ruizhong Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Designs, Codes and Cryptography*, 26(1-3):97–110, 2002.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In Janos Simon, editor, *Proceedings of the 20th annual ACM Symposium on Theory of Computing (STOC'88)*, pages 1–10. ACM, 1988.

[BS05] Boaz Barak and Amit. Sahai. How to play almost any mental game over the net-concurrent composition via super-polynomial simulation. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 543–552. IEEE, 2005.

[BSFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. *Advances in Cryptology–CRYPTO 2012*, pages 663–680, 2012.

[BT07] Anne Broadbent and Alain Tapp. Information-theoretic security without an honest majority. In *Advances in Cryptology–ASIACRYPT 2007*, pages 410–426. Springer, 2007.

[BT08] A. Broadbent and A. Tapp. Information-theoretically secure voting without an honest majority. In *Proceedings of the IAVoSS Workshop On Trustworthy Elections (WOTE'08)*, 2008. Cryptology ePrint Archive: Report 2008/266.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.

[Can06] Ran Canetti. Security and composition of cryptographic protocols: a tutorial (part i). *ACM SIGACT News*, 37(3):67–92, 2006.

[CC00] Christian Cachin and Jan Camenisch. Optimistic fair secure computation. In *Advances in Cryptology.Crypto 2000*, pages 93–111. Springer, 2000.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.

[CCM98] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 493–502. IEEE, 1998.

[CDD+99a] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In

Jacques Stern, editor, *Proceedings of the 18th Annual International Conference on the Theory and Applications of Cryptographic Technique (EUROCRYPT 99)*, pages 311–326. Springer, 1999.

[CDD+99b] Ronald Cramer, Ivan Damgård, Stefen Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology-EUROCRYPT 99*, pages 311–326. Springer, 1999.

[CDM00] Ronal Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Advances in Cryptology-EUROCRYPT 2000*, pages 316–334. Springer, 2000.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper B Nielsen. *Multiparty computation from threshold homomorphic encryption*. Springer, 2001.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology-CRYPTO 94*, pages 174–187. Springer, 1994.

[CHK+11] Jan Camenisch, Kristiyan Haralambiev, Markulf Kohlweiss, Jorn Lapon, and Vincent Naessens. Structure preserving cca secure encryption and applications. In *Advances in Cryptology–ASIACRYPT 2011*, pages 89–106. Springer, 2011.

[CK88] Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 42–52. IEEE, 1988.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global crs. In *Public-Key Cryptography–PKC 2013*, pages 73–88. Springer, 2013.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 494–503. ACM, 2002.

[CMW05] Claude Crépeau, Kirill Morozov, and Stefan Wolf. Efficient unconditional oblivious transfer from almost any noisy channel. In *Security in Communication Networks*, pages 47–59. Springer, 2005.

[Cré87] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In *Advances in Cryptology-CRYPTO 87*, pages 350–354. Springer, 1987.

[Cré97] Claude Crépeau. Efficient cryptographic protocols based on noisy channels. In *Advances in Cryptology-EUROCRYPT 97*, pages 306–317. Springer, 1997.

[CS91] Claude Crépeau and Miklós Sántha. On the reversibility of oblivious transfer. In *Advances in Cryptology-EUROCRYPT91*, pages 106–113. Springer, 1991.

[CT12] Thomas Cover and Joy Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[CvdGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. *Advances in Cryptology-CRYPT0 95*, pages 110–123, 1995.

[DBV03] Annalisa De Bonis and Ugo Vaccaro. Constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. *Theoretical Computer Science*, 306(1):223–243, 2003.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Theory of Cryptography*, pages 54–74. Springer, 2012.

[DFMS04] Ivan Damgård, Serge Fehr, Kirill Morozov, and Louis Salvail. Unfair noisy channels and oblivious transfer. *Theory of Cryptography*, pages 355–373, 2004.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520. IEEE Computer Society, 2010.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical covertly secure mpc for dishonest majority–or:

Breaking the spdz limits. In *Computer Security–ESORICS 2013*, pages 1–18. Springer, 2013.

[DKRS06] Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *Advances in Cryptology-CRYPTO 2006*, pages 232–250. Springer, 2006.

[DKS99] Ivan Damgård, Joe Kilian, and Louis Salvail. On the (im) possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Advances in Cryptology-EUROCRYPT99*, pages 56–73. Springer, 1999.

[DM99] Yevgeniy Dodis and Silvio Micali. Lower bounds for oblivious transfer reductions. In *Advances in Cryptology-EUROCRYPT99*, pages 42–55. Springer, 1999.

[DNRT12] Bernardo David, Ryo Nishimaki, Samuel Ranellucci, and Alain Tapp. General constructions of efficient multi-party computation. *Unpublished manuscript*, 2012.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.

[EFF85] Paul Erdös, Peter Frankl, and Zoltán Füredi. Families of finite sets in which no set is covered by the union ofr others. *Israel Journal of Mathematics*, 51(1-2):79–89, 1985.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[Eve83] Shimon Even. A protocol for signing contracts. *ACM SIGACT News*, 15(1):34–39, 1983.

[FGH+02] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam Smith. Detectable Byzantine agreement secure against faulty majorities. In *Proceedings of the 21st annual Symposium on Principles of Distributed Computing (PODC'02)*, pages 118–126. ACM, 2002.

[FHHW03] Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multi-party computation. In Eli Biham, editor, *Proceedings of the 22nd Annual International Conference on the Theory and Applications of Cryptographic Technique (EUROCRYPT'03)*, pages 51–67. Springer, 2003.

[FHM98] Matthias Fitzi, Martin Hirt, and Ueli Maurer. Trading correctness for privacy in unconditional multi-party computation. In Hugo Krawczyk, editor, *Proceedings of the 18th Annual International Cryptology Conference (CRYPTO'98)*, pages 121–136. Springer, 1998.

[FJN+13] Tore Kasper Frederiksen, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Minilego: Efficient secure two-party computation from general assumptions. In *Advances in Cryptology–EUROCRYPT 2013*, pages 537–556. Springer, 2013.

[FLM11] Marc Fischlin, Benoit Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. *Advances in Cryptology–ASIACRYPT 2011*, pages 468–485, 2011.

[FMR96] Michael Fischer, Silvio Micali, and Charles Rackoff. A secure protocol for the oblivious transfer. *Journal of Cryptology*, 9(3):191–195, 1996.

[FN12] Tore Kasper Frederiksen and Jesper Buus Nielsen. Fast and maliciously secure two-party computation using the gpu. Technical report, Cryptology ePrint Archive: Report 2013/046, 2012.

[FN13] Tore Kasper Frederiksen and Jesper Buus Nielsen. Fast and maliciously secure two-party computation using the gpu. In *ACNS*, pages 339–356, 2013.

[FNP04] Michael Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology-EUROCRYPT 2004*, pages 1–19. Springer, 2004.

[Fre12] Tore Kasper Frederisken. Optimizing actively secure two-party computations. Master's thesis, Aarhus Universit, 2012.

[FS90] U. Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426. ACM, 1990.

[GH08] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. *Advances in Cryptology-ASIACRYPT 2008*, pages 179–197, 2008.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *Journal of Cryptology*, 9(3):167–189, 1996.

[GL07] Jens Groth and Steve Lu. A non-interactive shuffle with pairing based verifiability. *Advances in Cryptology–ASIACRYPT 2007*, pages 51–67, 2007.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304. ACM, 1985.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[GMS08] V. Goyal, Payman Mohassel, and Adam Smith. Efficient two party and multi party computation against covert adversaries. *Advances in Cryptology–EUROCRYPT 2008*, pages 289–306, 2008.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 174–187. IEEE, 1986.

[GMW91] Oded. Goldreich, Silvio. Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. *Advances in Cryptology-CRYPTO 2006*, pages 97–111, 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. *Advances in Cryptology-EUROCRYPT 2006*, pages 339–358, 2006.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. *Advances in Cryptology–ASIACRYPT 2007*, pages 164–180, 2007.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. *Advances in Cryptology–EUROCRYPT 2008*, pages 415–432, 2008.

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth-Sahai proofs revisited. In *PKC'10*, volume 6056 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 2010.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. Ot-combiners via secure computation. In *Theory of Cryptography*, pages 393–411. Springer, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HK02] Te Sun Han and Kingo Kobayashi. *Mathematics of information and coding*, volume 203. AMS Bookstore, 2002.

[HKE13a] Yan Huang, Jonathan Katz, and Dave Evans. Efficient secure two-party computation using symmetric cut-and-choose. *IACR Cryptology ePrint Archive*, 2013:81, 2013.

[HKE13b] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *Advances in Cryptology–CRYPTO 2013*, pages 18–35. Springer, 2013.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Advances in Cryptology–CRYPTO 2011*, pages 132–150. Springer, 2011.

[HR07] iftach. Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 2007.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Theory of Computing and Systems, 1997., Proceedings of the Fifth Israeli Symposium on*, pages 174–183. IEEE, 1997.

[IKLP06a] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 2006.

[IKLP06b] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In Cynthia Dwork, editor, *Proceedings of the 26th Annual International Cryptology Conference (CRYPTO'06)*, pages 483–500. Springer, 2006.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi. Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology-CRYPTO 2003*, pages 145–161. Springer, 2003.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and J. Wullschleger. Constant-rate oblivious transfer from noisy channels. *Advances in Cryptology–CRYPTO 2011*, pages 667–684, 2011.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Extracting correlations. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 261–270. IEEE, 2009.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit. Sahai. Founding cryptography on oblivious transfer–efficiently. *Advances in Cryptology–CRYPTO 2008*, pages 572–591, 2008.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. *Advances in Cryptology-EUROCRYPT 2007*, pages 97–114, 2007.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 316–324. ACM, 2000.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero knowledge proofs. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 474–479. IEEE, 1989.

[KS64a] W Kautz and Roy Singleton. Nonrandom binary superimposed codes. *Information Theory, IEEE Transactions on*, 10(4):363–377, 1964.

[KS64b] W Kautz and Roy Singleton. Nonrandom binary superimposed codes. *Information Theory, IEEE Transactions on*, 10(4):363–377, 1964.

[KS08] Vladmir Kolesnikov and Thomas. Schneider. Improved garbled circuit: Free xor gates and applications. *Automata, Languages and Programming*, pages 486–498, 2008.

[KSS12] Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. Billion-gate secure computation with malicious adversaries. In *Proceedings of the 21st USENIX conference on Security symposium*, pages 14–14. USENIX Association, 2012.

[KSV07] Mehmet Kiraz, Berry Schoenmakers, and José. Villegas. Efficient committed oblivious transfer of bit strings. *Information Security*, pages 130–144, 2007.

[KW13a] Liina Kamm and Jan Willemson. Secure floating-point arithmetic and private satellite collision analysis. 2013.

[KW13b] Liina Kamm and Jan Willemson. Secure floating-point arithmetic and private satellite collision analysis. 2013.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1219–1234. ACM, 2012.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the ddh assumption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 446–466. Springer, 2011.

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. *IACR Cryptology ePrint Archive*, 2013:79, 2013.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The ips compiler: optimizations, variants and concrete efficiency. *Advances in Cryptology–CRYPTO 2011*, pages 259–276, 2011.

[LP04] Yehuda Lindell and Benny Pinkas. A proof of yao.s protocol for secure two-party computation. In *Electronic Colloquium on Computational Complexity*, volume 11, pages 607–620, 2004.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology-EUROCRYPT 2007*, pages 52–78. Springer, 2007.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of yaos protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *Theory of Cryptography*, pages 329–346. Springer, 2011.

[LP12] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of cryptology*, 25(4):680–722, 2012.

[LRM10] Christoph Lucas, Dominik Raub, and Ueli Maurer. Hybrid-secure MPC: Trading information-theoretic robustness for computational privacy. In *Proceedings of the 29th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'10)*, pages 219–228. ACM, 2010.

[MQU07] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. *Theory of Cryptography*, pages 41–60, 2007.

[MR13] Payman Mohassel and Ben Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. *IACR Cryptology ePrint Archive*, 2013:51, 2013.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, 4(2):151–158, 1991.

[NNOB12] Jesper Buus Nielsen, Peter Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. *Advances in Cryptology–CRYPTO 2012*, pages 681–700, 2012.

[NO09] Jesper Buus. Nielsen and Claudio Orlandi. Lego for two-party secure computation. *Theory of Cryptography*, pages 368–386, 2009.

[NOIMQ03] Anderson CA Nascimento, Akira Otsuka, Hideki Imai, and Joern Mueller-Quade. Unconditionally secure homomorphic pre-distributed commitments. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 87–97. Springer, 2003.

[NP99a] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254. ACM, 1999.

[NP99b] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *Advances in CryptologyCRYPTO99*, pages 573–590. Springer, 1999.

[OGW08] Silvio Micali Oded Goldreich and Avi Widgerson. How to play any mental game - a completness theorem for protocols with honest majority. *Distributed Computing and Networking*, pages 304–309, 2008.

[Orw49] George Orwell. *1984*. Erich Fromm, New York, 1949.

[OW05] Erik Ordentlich and Marcelo J Weinberger. A distribution dependent refinement of pinsker's inequality. *Information Theory, IEEE Transactions on*, 51(5):1836–1840, 2005.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology-CRYPTO 91*, pages 129–140. Springer, 1992.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 242–251. ACM, 2004.

[PSSW09] Benny Pinkas, Thomas. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. *Advances in Cryptology–ASIACRYPT 2009*, pages 250–267, 2009.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. *Advances in Cryptology–CRYPTO 2008*, pages 554–571, 2008.

[Rab81] Michael O Rabin. How to exchange secrets by oblivious transfer. Technical report, Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[RT13] Samuel Ranellucci and Alain Tapp. Efficient secure two-party computation from black-box primitives via faulty cut-and-choose ot. *IACR Cryptology ePrint Archive*, 2013:99, 2013.

[RTWW11] Samuel Ranellucci, Alain Tapp, Severin Winkler, and Jürg Wullschleger. On the efficiency of bit commitment reductions. In *Advances in Cryptology–ASIACRYPT 2011*, pages 520–537. Springer, 2011.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, nov 1979.

[Sha92] Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

[Spe28] Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.

[SS11] Abhi Shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. *Advances in Cryptology–EUROCRYPT 2011*, pages 386–405, 2011.

[SSR08] Bhavani Shankar, Kannan Srinathan, and C.Pandu Rangan. Alternative protocols for generalized oblivious transfer. *Distributed Computing and Networking*, pages 304–309, 2008.

[SWZ00] Douglas Stinson, Ruizhong Wei, and Lie Zhu. Some new bounds for cover-free families. *Journal of Combinatorial Theory, Series A*, 90(1):224–234, 2000.

[Tas11] Tamir Tassa. Generalized oblivious transfer by secret sharing. *Designs, Codes and Cryptography*, 58(1):11–21, 2011.

[Tho09] Mikkel Thorup. String hashing for linear probing. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 655–664. Society for Industrial and Applied Mathematics, 2009.

[Wie83] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983.

[WNI03] Andreas Winter, Anderson Nascimento, and Hideki Imai. Commitment capacity of discrete memoryless channels. In *Cryptography and Coding*, pages 35–51. Springer, 2003.

[Wul07] Jürg Wullschleger. Oblivious-transfer amplification. *Advances in Cryptology-EUROCRYPT 2007*, pages 555–572, 2007.

[Wul09] Jürg Wullschleger. Oblivious transfer from weak noisy channels. In *Theory of Cryptography*, pages 332–349. Springer, 2009.

[WW05] Stefan Wolf and Jürg Wullschleger. New monotones and lower bounds in unconditional two-party computation. In *Advances in Cryptology-CRYPTO 2005*, pages 467–477. Springer, 2005.

[WW10] Severin Winkler and Jürg Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In *Advances in Cryptology–CRYPTO 2010*, pages 707–723. Springer, 2010.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.