UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Anonymity and independence in multiparty protocols

Permalink

https://escholarship.org/uc/item/5pm6t9rc

Author

Hevia, Alejandro

Publication Date

2006

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Anonymity and Independence in Multiparty Protocols

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Alejandro Hevia

Committee in charge:

Professor Daniele Micciancio, Chair
Professor Samuel R. Buss
Professor Mihir Bellare
Professor Adriano Garsia
Professor Russell Impagliazzo

2006

Copyright

Alejandro Hevia, 2006

All rights reserved.

The dissertation of Alejandro Hevia is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2006

To Carolina

TABLE OF CONTENTS

Signature Page iii

Dedication iv

Table of Contents v

List of Figures viii

List of Tables ix

Acknowledgements x

Vita, Publications, and Fields of Study xii

Abstract of the Dissertation xiv

1 General Introduction 1

1.1 Cryptography and Provable Security 1

1.2 Secure Multiparty Computation 4

1.2.1 Motivation and Background 4

1.2.2 On the importance of definitions 6

1.3 Anonymity and Independence of Inputs 6

1.4 Contributions 8

2 Preliminaries 11

2.1 Definitions 11

2.1.1 Notation 11

2.1.2 Probability Distributions and Algorithms 12

2.1.3 Asymmetric Encryption 13

2.2 Secure Multiparty Computation 15

2.2.1 Terminology and Background 15

2.2.2 Multiparty Setting: Building Blocks 19

2.2.3 Standalone Security 22

2.2.4 Universally Composable Security 24

3 Anonymous Communication 30
3.1 Introduction 30
3.1.1 Coping with information leaks 31
3.1.2 Strong, Formal Definitions 34
3.1.3 Comparing Notions 35
3.1.4 The Anonymity of Previous Protocols 37
3.1.5 Comparison with Previous Anonymity Notions 38
3.1.6 Discussion and Related work 39
3.2 Preliminaries 40
3.3 Security Notions 41
3.4 Relation between the notions 43
3.4.1 Implications under computational assumptions 44
3.4.2 Implications that require "Dummy Traffic" 52
3.4.3 Message Overhead and Optimality of the Transformations 56
3.5 On the Anonymity of Previous Protocols 61
3.5.1 Broadcast Networks 61
3.5.2 DC-nets or Anonymous Broadcast 69
3.5.3 MIX networks 71
3.6 Variants and Extensions 79

4 Simultaneous Broadcast 81
4.1 Introduction 81
4.1.1 Discussion and Related Work 85
4.2 Preliminaries 86
4.2.1 The Model 86
4.2.2 Multisender Broadcast 87
4.3 Simultaneous Broadcast: Notions of Independence 88
4.3.1 Chor, Goldwasser, Micali and Awerbuch's definition 88
4.3.2 Chor and Rabin's definition 90
4.3.3 Gennaro's definition 90

4.4 The Role of the Input Distributions . . . 91
4.4.1 Distributions for CR-Independence . . . 93
4.4.2 Distributions for G-Independence . . . 95
4.4.3 Distributions for Sb-Independence . . . 97
4.4.4 Relations between Distributions . . . 97
4.5 Implications and Separations . . . 97
4.5.1 Separations . . . 101
4.5.2 Feasibility of CR and G independence . . . 105

5 Universally Composable Simultaneous Broadcast . . . 106
5.1 Introduction . . . 106
5.1.1 The Need for Efficient Simultaneous Broadcast with Strong Security Guarantees . . . 106
5.1.2 New Results . . . 108
5.1.3 Discussion and Related Work . . . 108
5.1.4 Comparison with previous solutions . . . 111
5.2 Preliminaries . . . 112
5.3 Terminating VSS (UC-TVSS) . . . 113
5.3.1 Instantiating TVSS . . . 116
5.4 Adaptively secure VSS of Cramer et al. . . . 117
5.4.1 Information Checking Protocol . . . 117
5.4.2 The Verifiable Secret Sharing Protocol of Cramer et al. . . . 122
5.5 UC Simultaneous Broadcast (UC-SB) . . . 127
5.5.1 A Generic Construction of UC-SB from UC-TVSS . . . 129
5.6 Extensions . . . 133

A Appendix . . . 136
A.1 Alternative Characterization of Simultaneous Broadcast Notions . . . 136
A.1.1 Sb-Independence . . . 136
A.1.2 CR-Independence . . . 137
A.1.3 G-Independence . . . 139

Bibliography . . . 143

LIST OF FIGURES

2.1 The synchronous communication functionality $\mathcal{F}_{SYN}$ of Canetti 28
2.2 The Verifiable Secret Sharing (with Spooling) functionality $\mathcal{F}_{VSS}$. 29

3.1 Examples of communication patterns hidden by each anonymity notion . 35
3.2 Anonymity variants and their associated relations $R_{\mathbf{N}}$. 43
3.3 Relations among notions of anonymity 45
3.4 A simplified version of the WAR protocol 63

4.1 Implications and separations for Simultaneous Broadcast definitions . . . 85

5.1 The Terminating Verifiable Secret Sharing (with Spooling) functionality $\mathcal{F}_{TVSS}$ 115
5.2 Information Checking Protocol of Cramer et al., sub-protocol `Distr` . . . 119
5.3 Information Checking Protocol of Cramer et al., sub-protocol `AuthVal` . . 120
5.4 Information Checking Protocol of Cramer et al., sub-protocol `RevealVal` . 121
5.5 Protocols for *giving* and *revealing* IC-signatures 123
5.6 Adaptively secure VSS protocol of Cramer et al., Sharing Phase 124
5.7 Adaptively secure VSS protocol of Cramer et al., Reconstruction Phase . 125
5.8 The simultaneous broadcast functionality $\mathcal{F}_{SB}$ 128
5.9 Simultaneous Broadcast protocol in the $\mathcal{F}_{TVSS}$-hybrid model 130
5.10 The asynchronous simultaneous broadcast functionality $\mathcal{F}_{ASB}$. 134

LIST OF TABLES

3.1 Anonymity variants and their alternative names 33

ACKNOWLEDGEMENTS

Without a doubt, I will remember my years spent at UCSD as one of the best times in my life. There are many people to thank for that.

First and foremost, I want to thank my wife and love of my life, Carolina, for her love and support during all these years of graduate school. Without her, this thesis would not be possible.

I am grateful to my advisor, Daniele Micciancio, who guided my development as researcher, for all his relentless support, guidance, and patience. I learnt so much from him, from how to be a researcher up to how to overcome hard times in life. Among many things, he taught me how to ask the "right questions" and how pursue them, to critically evaluate my own work, and to strive for excellence. Thank you, Daniele, for encouraging me to be the best I can be.

I also thank my other committee members: Professors Sam Buss, Mihir Bellare, Adriano Garsia, and Russell Impagliazzo for agreeing to supervise this thesis.

I am grateful to my friend and colleagues, former and present members of the Cryptography and Security Lab at UCSD: Michel Abdalla, Jee Hea An, Alexandra (Sasha) Boldyreva, Anand Desai, Marc Fischlin, Matthew Hohlfeld, Eike Kiltz, Tadayoshi (Yoshi) Kohno, Anton Mityagin, Sara Miner More, Chanathip (Meaw) Namprempre, Sir Gregory Neven, Saurabh Panjwani, Adriana Palacio, Tom Ristenpart, Haixia Shi, Sarah Shoup, Bogdan Warinschi, and Chong Zhang together with the "honorary members" of the lab: Nuno Bandeira, Yi-Kai Liu, Vadim Lyubashevsky, and Barath Raghavan. My sincere thanks for their helpful and enjoyable discussions, comments, insights, and being wonderful people to be around while studying, doing research, or having fun. I will not forget our time together or our long discussions at the Mandeville coffee cart.

I thank to people I have the honor to call co-authors: Daniele Micciancio, Flavio Junqueira, Ranjita Bhagwan, Keith Marzullo, Geoff Voelker Craig Gentry, Ravi Jain, Toshiro Kawahara, Zulfikar Ramzan, Anand Desai, Lisa Yin, and Marcos Kiwi.

I also want to thank my other friends from UCSD to whom I have owned so

much during my grad school years. Among them, Marvin McNett, and Flavio Junqueira for being extraordinary friends and colleagues, and their continuous help and support. Similarly, I must express my sincere gratitude to all the staff at the Computer Science Department at UCSD, specially to Julie Conner who not only is known to have magical powers to solve every administrative problem I ever had as grad student, but also happens to be the friendliest person in the world. Last but not least, I would not have reach the point where this thesis could have been written without the love and support of my close friends in San Diego: Todd Clements, Louise Chen, David and Cheryl Clements; Scott and Michelle Finkel; Jeff Conte, Michelle Dean; Jason and Amy Flowers; and Dave and Betty Little. Thanks for all your help, counsel, and friendship, specially for making me feel I belong in San Diego. I am truly happy and honored I can call you friends.

Finally, I would like to thank my parents whose continuous encouragement, love and support made possible y achievements. In particular, I would like to thank my mother, for being my number one fan; my father Francisco, for reminding me that life is meant to enjoy it and share it with the loved ones; and to my father Jorge, who taught me to love math and science, and show me the path to excellence. To all my family and friends in Chile, thank you for your love and support.

The material in Chapter 3 is from "An Indistinguishability-based Characterization of Anonymous Channels", joint work with Daniele Micciancio, currently submitted for publication. Most of Chapter 4 is a reprint from the material appearing in "Simultaneous Broadcast Revisited," with Daniele Micciancio, in the proceedings of 24th ACM Symposium on Principles of Distributed Computing (PODC 2005), Marcos Kawazoe Aguilera, James Aspnes Eds., ACM Press, 2005. Finally, most of Chapter 5 is a reprint from the material appearing in "Universally Composable Simultaneous Broadcast", in the proceedings of the 5th Security and Cryptography for Networks (SCN 2006), Springer-Verlag, 2006.

VITA

1973	Born, Viña del Mar, Chile
1998	B.A., University of Chile, Chile
2002	M.Sc. University of California, San Diego
2006	Ph.D. University of California, San Diego

PUBLICATIONS

A. Hevia, "Universally Composable Simultaneous Broadcast", In proceedings of the 5th Security and Cryptography for Networks (SCN 2006), LNCS, Springer-Verlag, 2006.

A. Hevia and D. Micciancio, "Simultaneous Broadcast Revisited," In 24th Symposium on Principles of Distributed Computing (PODC 2005) Proceedings, ACM Press, 2005.

F. Junqueira, R. Bhagwan, A. Hevia, K. Marzullo, and G. M. Voelker, "Surviving Internet Catastrophes," In Proceedings of the USENIX Annual Technical Conference, Anaheim, California, USA, 2005.

C. Gentry, A. Hevia, R. Jain, T. Kawahara, and Z. Ramzan, "End-to-End Security in the Presence of Intelligent Data Adapting Proxies: the Case of Authenticating Transcoded Streaming Media," In IEEE Journal on Selected Areas of Communications, Special Issue on Intelligent Services and Applications in Next Generation Networks, Vol. 23 n.2, IEEE Press, Feb. 2005.

A. Hevia and D. Micciancio, "The provable security of Graph-Based One-Time Signatures and extensions to algebraic signature schemes," In Advances in Cryptology - Asiacrypt 2002 Proceedings, pages 379-396, LNCS 2501, Springer-Verlag, 2002.

A. Desai, A. Hevia, and Y. L. Yin, "A Practice-Oriented Treatment of Pseudorandom Number Generators," In Advances in Cryptology - Eurocrypt 2002 Proceedings, LNCS 2332, pages 368-383, Springer-Verlag, 2002.

A. Hevia and M. Kiwi, "Electronic Jury Voting Protocols," In Theoretical Computer Science Volume 321, Issue 1, 16 June 2004, Pages 73-94. Elsevier Science, 2004.

SUBMITTED PUBLICATIONS

A. Hevia and D. Micciancio, "An Indistinguishability-based Characterization of Anonymous Channels", Submitted for publication.

FIELDS OF STUDY

Studies in Cryptography
Professor Daniele Micciancio

ABSTRACT OF THE DISSERTATION

Anonymity and Independence in Multiparty Protocols

by

Alejandro Hevia

Doctor of Philosophy in Computer Science

University of California San Diego, 2006

Professor Daniele Micciancio, Chair

Electronic voting systems, chat rooms, and electronic auctions are well-known examples of systems that involve many people interacting at the same time in order to achieve a goal: cast a vote, discuss political ideas, and bid for an item. Yet, some security goals of these *multi-party protocols* have not been carefully examined as they should. For example, how different is protecting the anonymity of the sender – as in an electronic election – versus protecting the anonymity of both the sender and the receiver – as in a political chat room? Or, can online surveys guarantee that participants do not influence each other's answers? This dissertation studies two of these properties: anonymity and independence of inputs.

We first revisit the problem of *anonymous communication*, in which users wish to send messages to each other without revealing their identities. We propose a novel framework to organize and compare anonymity definitions. In this framework, we present simple and practical definitions for anonymous channels in the context of computational indistinguishability. The notions capture the intuitive properties of several known types of anonymous channels, and also model practical scenarios where information is leaked in the system. We also compare these notions showing how stronger notions can be built from weaker ones in a provably optimal way. With these tools, we revisit the security of previous popular anonymous channels protocols.

In applications where multiple senders can broadcast messages at the same time (like electronic auctions or elections), it is often important to enforce the *simultaneous* transmission of messages, so that no sender can decide its broadcast message based on the values broadcast by other players. In the second part of the dissertation, we study the definitions of independence (or *Simultaneous Broadcast*) proposed in the literature, obtaining a full characterization of their applicability and relative strength. In particular, we show that the latest definition, under which the most (round) efficient solution was proven secure, is strictly weaker than the previous definitions. We thus reopen the problem of either proving the efficient solution satisfies the strongest definition of security or finding an alternative efficient protocol that does so.

We conclude by addressing this last problem. We present a definition of simultaneous broadcast under the Universally Composable framework, and show that this stronger notion can be achieved by a computationally efficient, constant-round construction. Our construction builds on an existent adaptive verifiable secret sharing scheme, and does not resort to generic zero-knowledge proofs or commitment schemes present in previous solutions.

1

General Introduction

1.1 Cryptography and Provable Security

In the past three decades, our society has become increasingly dependent on information stored in *digital* form. This information is usually accessed and used by many different entities (computers), from many different geographical locations over large networks (like the Internet). Examples of this include electronic voting systems, online banking systems, medical information systems, electronic auctions, and online tax filing, to name a few. There are many potential benefits from having these distributed solutions; for example, electronic voting schemes may allow voters to cast ballots from remote locations easily, and may help governments to have fast and inexpensive elections. Of course, moving information to the digital world raises several security concerns, ranging from how to protect the privacy of the data flowing on the network (eg. each voter's candidate), up to how to assure the participating entities do not exploit subtle timing interactions of the network to compromise the goals of the system (eg. some misbehaving government official finding the identities of voters who *did not* vote for a given candidate).

Cryptography is the field that studies techniques to understand some of the above problems, and designs tools to implement solutions for them. Historically, two basic goals of cryptography have been data privacy and authentication. The former aims to allow two parties to exchange data over a communication channel (through the mail, a

telephone line, or wireless network link) so the actual content sent on the communication line is unintelligible and meaningless to any curious eavesdropper of the line except the intended receiver. This problem has been around for centuries (Julius Caesar employed some basic letter substitution techniques to hide messages of military importance) and its solution, encryption, has been extensively study. Data authentication, another basic goal of cryptography, allows a sending party to convince a receiving party that the data received indeed comes from the sending party – so, for example, a bank can be sure client Bob's remote request to pay a large amount of money to Alice indeed is an authorization that comes from Bob and it was not maliciously made by Alice herself.

Even though data privacy and authentication tools are crucial components to any secure system like those mentioned before, the fact those systems must work over complex, dynamic distributed environments, with entities accessing remote data at different times (eg. concurrently) and from different locations raises new issues. For instance, systems to remotely access information on sensitive diseases (HIV, cancer) may need to assure their users' *anonymity* – the identities of the users of the system are not revealed to anyone, not even to a curious insider (the system manager). Another case, for example, would be where the auctioneer of an online auction system may wish to have guarantees that malicious bidders cannot "correlate" their bids to those of honest bidders, so the final price of items reflect the bidders true valuation (and nothing less[1]). In other words, the auctioneer would like to enforce the *independence of inputs* for all bidders. In this thesis, we show how cryptographic techniques can help to further understand the subtleties involved in identifying the appropriate security goals such multi-party system must seek, as well as designing efficient implementations of them.

But how do we know if a system or protocol is "secure"? As cryptographers have known for long time, achieving security is hard since, at first, it seems to require proving a negative: that no malicious entity (adversary) exists for the system. Instead, cryptographic protocols are often designed by trial and error: a scheme is assumed secure if it is "believed" to be secure – more precisely, if it has not been broken. This approach is highly unreliable, as often the best designer's or evaluator's intuition may be

[1] It would be extremely unfair (and probably would not make the auctioneer happy) if a thief could somehow manage to set the winning bid to only one cent more than the second-highest bid, even if the thief does not know the value of his/her bid!.

flawed, and insecure protocols may be deployed and widely employed a long time before serious security flaws or attacks are discovered. Serious breaks on the wireless protocol WEP [BGW01], the Diebold electronic voting system [KSRW04], or the GSM cellphone encryption protocol [GBW98] are well-know examples of this.

Perhaps surprisingly, the above approach to protocol design is not the only one. It is possible to *prove* that a protocol resists some attacks, *under the assumption some other well-known problem is hard to solve or some primitive is hard to attack*. This technique, the *provable security* approach, was first proposing by Goldwasser and Micali [GM84] and it works by mathematically proving that any attack on the protocol in question can be translated into an attack on some other (often smaller, more studied) cryptographic primitive or mathematical problem. This "translation" is called the *security reduction*. In order to be meaningful, a reduction-based proofs require the specification of a security model, which includes the description of the adversarial capabilities (what the adversary can do, including the available resources), and the security goal (what the adversary should not be able to do). Proving the security of a protocol in this framework is done by contradiction: *if* an adversary with the specified capabilities violates the security goal, then some well-known assumption no longer holds. More precisely, it involves showing how any adversarial algorithm which violates the security goal using the specified capabilities (eg. uses at most *polynomial* time, eavesdrops a certain number of messages, or can access local private data of at most a certain number of honest parties in the system) can be transformed into another adversarial algorithm that violates another well-known cryptographic problem or primitive. Therefore, if the latter well-known problem or primitive is believed to be secure, then no such adversary for the original protocol can exist in the first place. Examples of well-known problems are mathematical problems long conjectured to be hard to solve, like the hardness of factoring large numbers or the hardness of computing discrete logarithms in certain cyclic groups. Similarly, reductions to the hardness of breaking some properties of cryptographic primitives (pseudorandomness of block ciphers when seen as families of functions, for example) have also been used to justify the security of more complex protocols. As long as the security model captures meaningful threats to the protocol at hand the provable security approach provides very strong security guarantees: the protocol will not be broken

unless the hardness of the underlying problem does not hold, thus reducing our focus to the security of these underlying (usually extensively studied) problems.

1.2 Secure Multiparty Computation

1.2.1 Motivation and Background

A classical setting to analyze problems like encryption or data authentication involves two participants – one sender and one receiver – and one adversary – the eavesdropper or forger. Such setting has shown certainly adequate for these problems as it simplifies the security model, and in consequence, the security proofs. In contrast, systems like electronic voting protocols or electronic auctions explicitly involve many participants, and in addition an adversary able to compromise (break-into) some of these participants. For this reason, this last type of protocols have been often referred as *multi-party protocols*.[2] Loosely speaking, in multi-party protocols, several parties are often actively involved in the execution of the protocol (but perhaps not all actively involved at every moment). Multiparty protocols are interesting cryptographically as their analysis must explicitly take into account the *influence* of multiple participants with different degrees of participation into the security of the protocol. In general, this influence is often expressed in terms of each participant's actions (how parties can affect the protocol), or each participant's knowledge of private data or ability to gather information (how parties can obtain information from the protocol). All of this may add new complexities and new sources of threats. For example, a seemingly secure system may leak private information when executed in conjunction with some other system (that is, when *composed*[3]), or a multiparty system may compromising the anonymity of the participants because of some participant's *inaction* (not sending data out) may

[2] The distinction between "multi-party protocols" and "single sender/single receiver schemes" (like those implementing encryption and data authentication) is rather artificial as *schemes* are usually two party protocols which are a special case of multiparty protocols. Moreover, there are good benefits of casting encryption or signature schemes in a multiparty setting as many recent results have shown (e.g. [BBM00], [CK02]). Nonetheless, since the results of these thesis deal with problems (anonymity and independence of inputs) where the most interesting cases happen with more than two parties, we add the term "multiparty" to the protocols we discuss.

[3] A well-known example is the case of some zero-knowledge proof protocols [GMR89]. Some of them, despite being proven secure as standalone protocols, were not secure when executed in parallel [GK96].

help an eavesdropper to narrow it down the identity of a certain message's sender. Also, issues like execution synchronization (whether a global clock exists or not) or fault tolerance resilience (how the protocol behaves when participants unexpectedly stop or no longer follow the protocol) become a more challenging concern in the multiparty setting. Therefore, the security goals and adversarial capabilities required to carry on the provably secure approach become more subtle to characterize as capturing what "secure" means is not necessarily straightforward.

Fortunately, there has been extensive work on the subject, including on security models (see [Can05] and references therein) and concrete protocols. Regarding security models, the standard paradigm is to define security of a protocol in terms of *emulation* of an trusted party. The idea is that any protocol can be thought as computing a process (for example, a function) on some party inputs. Then, *if* we had a completely reliable party which is trusted by all, parties could simply hand all inputs out to the trusted party, which would perform the computation and return the outputs back to each party. The goal then would be that a secure protocol should be able to "emulate" this idealized computation by the mutually distrustful parties. The concrete meaning of emulation can vary, depending on the assumptions on the communication channels, or the type of tolerated adversarial behavior, for example. Some of these parameters are discussed in Section 2.2.1.

Interestingly, in the past thirty years, there have been proposed not only a variety of protocols for solving specific multiparty problems but also concrete protocols for solving *general* multiparty problems. Protocols that solve arbitrary problems – where the process to compute can be parameterized by an arbitrary function – were proposed in the seminal and independent work of: Goldreich, Micali and Widgerson [GMW87], Ben-Or, Goldwasser and Widgerson [BGW88] and Chaum, Crépeau and Damgård [CCD88]. The efficiency of these proposed solutions for general problems has been improved in recent years (eg. [CDD$^+$99, BTH06] among many others), although for some concrete problems like electronic voting or threshold cryptography, for example, more efficient solutions have come from specially tailored constructions. In terms of security analysis, more sophisticated security models for the problems have been proposed and further studied. Two important popular variants of these models, proposed by Canetti 2000

[Can00a] and Canetti 2001 [Can01b, Can05] are further discussed in Section 2.2.1.

1.2.2 On the importance of definitions

From the view point of provably security, it is hard to overstate the the importance of the security model: any security reduction can only capture the concerns and threats explicitly implied by the security goal and by the adversarial capabilities. Indeed, all bets are off for adversaries that do not fit the security model: the security reduction provide no guarantees such adversaries do not exist, even if the assumption on the hardness of the base problem happens to be true. In other words, the security model can be seen as providing the *meaning* of the word "security" for any protocol. Therefore, understanding the practical applicability and limitations of the security model for a given problem is as critical as developing a secure implementation. The flip side of this consideration is that some *too practical* threats may be hard to include in the proof: attacks involving key compromises, worms leveraging on software vulnerabilities, hardware failures, or some denial-of-service attacks are often not captured in security proofs as factoring them into the adversarial capabilities can obscure some other most important threats and complicates the model. Nonetheless, it seems the answer here is not that some threats and concerns do have to remain always beyond the guarantees of the security proof. Recent trends in this direction show that including more realistic adversarial capabilities and security goals may lead to protocols more secure in "real life" (e.g. [MR04]). We claim that plugging those capabilities into the model is not only necessary but critical to obtain meaningful security guarantees. In fact, we follow this approach in Chapter 3, by adding adversarial capability to select parameters like the amount of network traffic per party to model leakage of information in anonymity systems.

1.3 Anonymity and Independence of Inputs

In this dissertation we study two important goals in multiparty computation: anonymous communication and independence of inputs.

Anonymous communication allows users to send and receive messages without revealing their identities. There are well-known ways to obtain anonymity in the *non-digital* world: one could send stamped mail from a randomly selected post-office or could make phone calls from a payphone. In the digital world, things become much harder as each action or message sent in a digital form can be copied or traced if communication links are monitored (which nowadays is well-known to be feasible). Yet, anonymous communication plays an very important role in many (very legitimate) settings. For example, protecting "whistle blowers" or guaranteeing source confidentiality in crime tips, fostering online support groups (for "embarrassing" or socially stigmatized diseases like HIV, cancer or sexually transmitted diseases), protecting voter privacy in electronic voting, providing effective tools to implement security policies for managing patient health records, or even preventing (unfair) price discrimination.

As a cryptographic goal, anonymous communication is orthogonal to standard goals like confidentiality (secrecy of the *content* of the communication), or integrity of communicated data; clearly, by simply encrypting a message before sending it out in a regular communication channel, Bob does *not* hide the fact he is the sender. In fact, anonymous communication is one of the few applications which are *multiparty* by nature – all known techniques to solve the problem in the digital world (which do not somehow rely on specially trusted entities) involve participation of as many participants as possible. The obvious example is that no protocol can achieve meaningful anonymity for a system with only two participants. Therefore, sound solutions for the problem of anonymous communication face the same definitional subtleties and design challenges that several well-known multiparty protocols (like threshold encryption or distributed key generation) have face. Moreover, anonymous channels have been historically very hard to achieve. The reasons are often twofold. First, there are many practical details and realistic threats that are not adequately captured by the security models underlying previous anonymity-enabled constructions. And second, often the *meaning* of the different "flavors" of anonymity are rather informal or cast in definitions too weak to provide strong enough anonymity guarantees – at least strong enough as the guarantees commonly given by other more mature cryptographic primitives (like encryption). In this thesis, we study the anonymous communication problem, proposing security models

that address these two issues.

Another desirable property of multiparty protocols is independence of inputs. Loosely speaking, inputs provided by parties in a protocol are independent if no party can decide (change) his/her local input based on the inputs by the other (honest) parties. In some sense, this property captures the concept of "fairness" in how inputs from the parties are contributed into the protocol. Indeed, independence guarantees each party, once given an input (eg. on the onset of the computation, or as the output of past protocol executions), the party contributes this value as his/her true input to the protocol. An important scenario where this property appears is in broadcast channels, which allow parties to transmit messages to be received by all parties connected to a network. Broadcast channels are powerful, commonly used primitives in the design of a large fraction of multiparty protocols, as they provide a way to reliably transmit consistent data (and thus provide a method to guarantee data consistency across several parties). In applications where multiple senders can broadcast messages at the same time (e.g., when running in parallel many copies of a broadcast protocol with different senders), it is often important to enforce this *simultaneous* transmission of the messages, so that no sender can decide its broadcast message based on the values broadcast by the other players. This independence property plays a fundamental role in the secure multiparty computation protocol of [CGMA85] as well as many important applications (like contract bidding, coin flipping, and electronic voting schemes) where broadcast is employed. In this thesis, we study the *simultaneous broadcast* problem, which is perhaps the simplest but still useful context in which the independence problem appears. Our study explores the definitional challenges surrounding the formalization of the security model for this problem, as well as how to achieve efficient constructions in very strong models.

1.4 Contributions

First, in the context of anonymous communication, we propose a novel framework to organize and compare anonymity definitions. In this framework, we present simple and practical definitions for anonymous channels in the context of computational

indistinguishability. The notions seem to capture the intuitive properties of several types of anonymous channels [PK01] (eg. sender anonymity and unlinkability). We justify these notions by showing they naturally capture practical scenarios where information is unavoidably leaked in the system. Then, we compare the notions and we show they form a natural hierarchy for which we exhibit non-trivial implications. In particular, we show how to implement stronger notions from weaker ones using cryptography and dummy traffic – in a provably optimal way. With these tools, we revisit the security of previous anonymous channels protocols, in particular constructions based on broadcast networks [BIK+03], anonymous broadcast [Cha81, GJ04], and mix networks [Gro03, NSNK04]. Our results give generic, optimal constructions to transform known protocols into new ones that achieve the strongest notions of anonymity.

Second, in the context of independence of inputs in multiparty protocols, we study various definitions of independence proposed in the literature by Chor, Goldwasser, Micali and Awerbuch [CGMA85], Chor and Rabin [CR87] and Gennaro [Gen00], and prove implications and separations among them. In summary, we show that each definition (generalized to allow arbitrary input distributions) is characterized by a class of "achievable" input distributions such that there is a single protocol that simultaneously meets the definition for all distributions in the class, while for any distribution outside the class no protocol can possibly achieve the definition. When comparing sets of achievable distributions, the definition of Gennaro is the most stringent (followed by the Chor and Rabin one, and Chor, Goldwasser, Micali and Awerbuch as the most relaxed) in the sense that it is achievable for the smallest class of distributions. This demonstrates that the definitions of Gennaro, and Chor and Rabin are of limited applicability.

Then, we compare the definitions when restricted to achievable distributions. This time the results of our comparison rank the definitions in the opposite order, with the definition of Chor, Goldwasser, Micali and Awerbuch as the strongest one (followed by Chor and Rabin, and then Gennaro) in the sense that security according to the stronger definitions implies security according to the weaker ones. We also give examples showing that the implications are strict, i.e., there are input distributions such that a protocol can meet the weaker definition, but fail to satisfy the stronger. The separation between the definitions of Gennaro and Chor and Rabin is particularly strong, as we show

that there is a single protocol that is simultaneously secure according to Gennaro under any achievable input distribution, but does not satisfy the definition of Chor and Rabin for any non-trivial distribution. In particular, the separation holds for the special case of the uniform input distribution originally considered by the authors in their papers.

Finally, we explore the problem of Simultaneous Broadcast under Universally Composable (UC) security of Canetti [Can01b, Can05]. We give a definition of Simultaneous Broadcast in this framework, which is shown to imply all past definitions. We also show this notion can be achieved by a computationally efficient, constant-round construction for the problem (building on the verifiable secret sharing scheme of Cramer et al. [CDD$^+$99]) which is proven secure as long as a majority of the parties are honest. Our results rely on (and benefit from) capturing synchronous communication as a functionality within the UC model, as suggested by Canetti [Can05]. Indeed, we show that this approach of modeling synchronous communication can lead to better understanding of where synchronicity is needed, and also simpler constructions and proofs.

2

Preliminaries

In this chapter, we introduce notation and concepts used throughout this thesis.

2.1 Definitions

2.1.1 Notation

Let $\mathbb{N}$ be the set of natural numbers. We denote by $\{0,1\}^*$ the set of all binary strings of finite length. Given a string $w \in \{0,1\}^*$, we let $|w|$ denote the length of w. Let $k \in \mathbb{N}$ denote a security parameter.

We let 1^k denote the string consisting of concatenating bit "1" exactly k times. When k is the security parameter, we often give this string as input (parameter) to algorithms so we can measure their running time as a function of k, ie. the algorithm's the input length. We say an algorithm S *runs in polynomial time* on its input length (which is often equal to the security parameter) if there exists a polynomial q such that, on input $w \in \{0,1\}^*$, the algorithm makes at most $q(|w|)$ steps before halting. Before halting, the algorithm may or may not have output some value. We refer to these algorithms as *polynomial-time* algorithms; if a polynomial-time algorithm is randomized, we say it is a *probabilistic polynomial-time* (PPT) algorithm.

Let n be a positive integer, and $[n]$ denote the set $\{1,\ldots,n\}$. For any set

$S \subset [n]$ and any vector $\mathbf{x} = (x_1, \ldots, x_n)$, we denote by $\mathbf{x}_S$ the $|S|$-dimensional vector formed by the elements of $\mathbf{x}$ whose index are in S, that is, $\mathbf{x}_S = (x_i)_{i \in S}$. Also, let G and B two disjoint sets such that $G \cup B = [n]$ and $\mathbf{w}, \mathbf{z}$ two n-dimensional vectors. Then, we let $\mathbf{w}_G \sqcup \mathbf{z}_B$ denote the n-dimensional vector formed by combining the elements of $\mathbf{w}$ with indexes in G with the elements of $\mathbf{z}$ with indexes in B. When clear from context, we may drop the subindex G or B, as in $\mathbf{w}_G \sqcup \mathbf{z}$. In such case, by convention, we assume the coordinates for $\mathbf{z}$ are in the set $\overline{G} = [n] \setminus G$. When describing multiparty protocols, we let $P_{[n]}$ denote the set of all parties $\{P_1, \ldots, P_n\}$.

2.1.2 Probability Distributions and Algorithms

Probability Distributions, Ensembles and Classes of Distributions: A probability distribution $\mathcal{D}$ is a function from strings to non-negative reals such that $\sum_{x \in \{0,1\}^*} \mathcal{D}(x) = 1$. For any distribution $\mathcal{D}$ over $\{0,1\}^n$ we write $\mathbf{d} \stackrel{R}{\leftarrow} \mathcal{D}$ to denote the process of selecting an n-dimensional vector $\mathbf{d}$ from $\{0,1\}^n$ according to distribution $\mathcal{D}$. We also denote by $\mathcal{D}_B$, for any $B \subset [n]$, the distribution induced by selecting a vector in $\mathcal{D}$ and taking only the coordinates in set B. For simplicity, we write $\mathcal{D}_i$ instead of $\mathcal{D}_{\{i\}}$. We also extend the $\sqcup$ notation to distributions. Given two distributions $\mathcal{D}$ and $\mathcal{R}$ over n-bit strings, for any set $B \subset [n]$, we say an n-bit vector $\mathbf{x}$ is drawn from distribution $\mathcal{D}_B \sqcup \mathcal{R}_{\overline{B}}$ if $\mathbf{x}$ is formed by first drawing $\mathbf{x}_B$ from $\mathcal{D}_B$ and then drawing $\mathbf{x}_{\overline{B}}$ from $\mathcal{R}_{\overline{B}}$. Notice that for any distribution $\mathcal{D}$ and set B, $\mathcal{X} \stackrel{\text{def}}{=} \mathcal{D}_B \sqcup \mathcal{D}_{\overline{B}}$ is not necessarily equal to $\mathcal{D}$ since $\mathcal{X}_B$ is independent from $\mathcal{X}_{\overline{B}}$ while $\mathcal{D}_B$ and $\mathcal{D}_{\overline{B}}$ may be dependent.

A probability ensemble indexed by $\mathbf{N}$ is a sequence $\Delta = \{\mathcal{D}^{(k)}\}_{k \in \mathbf{N}}$ of probability distributions. For each value of the security parameter k, probability distribution $\mathcal{D}^{(k)}$ assigns positive probability only to n-bit strings. We sometimes abuse notation by using $\mathcal{D}^{(k)}$ to refer to the *random variable* that ranges over $\{0,1\}^n$ and that follows the corresponding distribution $\mathcal{D}^{(k)}$. As with distributions, given a probability ensemble $\mathcal{D} = \{\mathcal{D}^{(k)}\}$ and a set $B \subset [n]$, we let $\mathcal{D}_B = \{\mathcal{D}_B^{(k)}\}_{k \in \mathbf{N}}$ denote the ensemble consisting of the induced distributions $\mathcal{D}_B^{(k)}$. A class of probability ensembles (or simply, class of distributions) $\Phi = \{\Delta^{\langle \ell \rangle}\}_{\ell \in \mathbb{D}}$ is a collection of probability ensembles $\Delta^{\langle \ell \rangle}$ indexed by some (possibly uncountable) set $\mathbb{D}$.

ALGORITHMS AND THEIR PROBABILITIES: For any (probabilistic) algorithm A, $A(x)$ denotes the probability distribution of all possible outputs of running algorithm A on input x. If P is a predicate, A, B are (probabilistic) algorithms, and x, y are values, then $\Pr[\, a \leftarrow A(x),\ \ b \leftarrow B(y), \ldots :\ \ P(a, b, \ldots)\,]$ denotes the probability that predicate P on input $a, b, \ldots$ is true given that $a, b, \ldots$, are the output of the ordered execution of algorithm A on input x, B on input y, and so on. A function $\mu(k)$ is negligible in the security parameter k if there exists a constant $c > 0$ and infinitely many positive values of k such that $\mu(k) < k^{-c}$. A probability is overwhelming if it is larger than $1 - \mu(k)$ where $\mu(k)$ is a negligible function.

2.1.3 Asymmetric Encryption

We recall the standard definition of asymmetric encryption schemes.

SYNTAX: Let $k \in \mathbb{N}$ be the security parameter. An asymmetric (public-key) encryption scheme $\mathcal{AS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms:

- A randomized polynomial-time *key generation* algorithm $\mathcal{K}$. This algorithm takes as input 1^k and returns a pair (pk, sk) consisting of a public key pk and a corresponding secret key sk. In this case, we write $(pk, sk) \xleftarrow{R} \mathcal{K}(1^k)$.

- A randomized polynomial-time *encryption* algorithm $\mathcal{E}$. This algorithm takes as input a public key pk and a plaintext m, and returns a ciphertext c. For this operation, we write $c \xleftarrow{R} \mathcal{E}(pk, m)$. Sometimes, it may be necessary to precise the particular value of the random bits $r \in \{0,1\}^*$ used by algorithm $\mathcal{E}$ to compute the ciphertext c; in such cases, we write $c \xleftarrow{R} \mathcal{E}(pk, m; r)$ with the implicit convention r is drawn uniformly at random from the set of strings $\{0,1\}^\ell$, for some long enough $\ell \in \mathbb{N}$ (usually $\ell = s(k)$ for some fix polynomial depending only on the scheme $\mathcal{AS}$).

- A deterministic polynomial-time *decryption* algorithm $\mathcal{D}$. This algorithm takes as input a secret key sk and a ciphertext c and returns a plaintext m' or the special symbol $\perp$ indicating the ciphertext c was invalid. For this operation, we write $m' \leftarrow \mathcal{D}(sk, c)$.

Associated to each public key pk there is a *message space* $\mathrm{MsgSp}(pk)$ from which the plaintext m is drawn. We require that $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$ for all $m \in \mathrm{MsgSp}(pk)$ and all (pk, sk) output by $\mathcal{K}(1^k)$.

SECURITY: We present the standard definition of data privacy for asymmetric encryption schemes following the indistinguishability approach [GM84]. In terms of adversarial capabilities, there are two possibilities: chosen-plaintext attacks and chosen-ciphertext attacks. A stateful adversary A runs in two stages, the "find" stage and the "guess" stage. In the first stage, given a public key, A outputs two equal-length messages m_0, m_1 together with some state information s. Then, in the "guess" stage, A gets a challenge ciphertext y which is the encryption of either m_0 or m_1 (this choice depends on a bit b which can be seen as chosen at random), and A must *guess* which message was chosen. For the case of chosen-ciphertext attacks [RS91], the adversary is also given access to a decryption oracle $\mathcal{D}(sk, \cdot)$ which A is free to evaluate at any value as long as the oracle is not queried on the challenge ciphertext c.

Let $\mathcal{AS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Let $b \in \{0, 1\}$, $k \in \mathbb{N}$ be the security parameter, and A_{cpa}, A_{cca} be adversaries where the latter has access to an oracle. Consider the following two experiments, parameterized by bit b:

Experiment $\mathbf{Exp}^{ind-cpa-b}_{\mathcal{AS},A_{cpa}}(k)$	**Experiment** $\mathbf{Exp}^{ind-cca-b}_{\mathcal{AS},A_{cca}}(k)$
$(pk, sk) \xleftarrow{R} \mathcal{K}(1^k)$	$(pk, sk) \xleftarrow{R} \mathcal{K}(1^k)$
$(m_0, m_1, s) \leftarrow A_{cpa}(\text{"find"}, pk)$	$(m_0, m_1, s) \leftarrow A_{cca}^{\mathcal{D}(sk,\cdot)}(\text{"find"}, pk)$
$y \leftarrow \mathcal{E}(pk, m_b)$	$y \leftarrow \mathcal{E}(pk, m_b)$
$g \leftarrow A_{cpa}(\text{"guess"}, y, s)$	$g \leftarrow A_{cca}^{\mathcal{D}(sk,\cdot)}(\text{"guess"}, y, s)$
return g	**return** g

The adversary $A \in \{A_{cpa}, A_{cca}\}$ is *legitimate* if it only outputs equal length plaintexts $m_0, m_1 \in \mathrm{MsgSp}(pk)$ and if $A = A_{cca}$, it does not query the oracle with y in the "guess" stage. For $atk \in \{cpa, cca\}$, encryption scheme $\mathcal{AS}$ achieves *indistinguishability against chosen plaintext attack* or simply IND-CPA (respectively *indistinguishability against chosen ciphertext attack* or simply IND-CCA) if the quantity

$$\mathbf{Adv}^{ind-atk}_{\mathcal{AS},A}(k) \stackrel{\text{def}}{=} \Pr\left[\mathbf{Exp}^{ind-atk-1}_{\mathcal{AS},A_{atk}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{ind-atk-0}_{\mathcal{AS},A_{atk}}(k) = 1\right]$$

is negligible in k for any feasible (PPT in k) adversary A_{atk}.

2.2 Secure Multiparty Computation

2.2.1 Terminology and Background

Very broadly, a multiparty protocol is a distributed algorithm in which two or more parties or participants compute a *functionality*, a random process that maps m inputs into m outputs. For purposes of this thesis, we consider $n \in \mathbb{N}$ parties, $m = n$, where the inputs to the process are the local inputs to the parties and the outputs are the local outputs to the parties. We consider a system of n participants (also called *parties* or *players*) whose identities are denoted by $P_1, P_2, \ldots, P_n$. We often let n denote either a fixed constant or a quantity that may depend polynomially on the security parameter k (eg. $n = q(k)$ for some fixed polynomial q). Formally, parties are modeled as probabilistic, polynomial-time (PPT) interactive Turing machines, although the exact characterization is not crucial for this thesis, as long as each party is polynomially bounded.[1] All parties belong to a network where they can communicate via messages. As standard in cryptography, we assume there is an adversary, modeled by an arbitrary non-uniform PPT algorithm. Parties execute a prescribed program (specified by the protocol) but some of them can be "influenced" by the adversary– how and the degree of influence is explained below. Parties which strictly follow their program are called *honest* (or uncorrupted), otherwise are called *corrupted* (or dishonest).

There are several ways to define the security of a protocol. Here we describe the *simulation paradigm* [Gol04]. As mentioned above, the particular functionality computed by the parties is described by the random process parties compute. But how do we know if an execution of a protocol computes a process "as it should"? To avoid a circular definition, it is useful to consider the abstraction in which the random process is computed by an external entity trusted by all participants, which receives all inputs from each party, computes the random process, and hands out the outputs for all. Then, we can measure security as how well the execution of a protocol in the presence of some disruptive entity (adversary) compares with an execution of the *ideal* process (where the trusted entity exists). If whatever a reasonable adversary (see below) can do in the

[1] The issue of capturing in a model the requirement parties are bounded to polynomial amount of work/time is non-trivial, particularly in the context of concurrent execution. General and sensible definitions are beyond the scope of this thesis. See a discussion on [Can05].

execution of the *real* multiparty protocol, a reasonable adversary can do in the ideal process. We further elaborate on this idea below and in Sections 2.2.3 and 2.2.4.

Below we discuss some parameters usually considered in defining security models in the literature.

ADVERSARIES: In order to capture meaningful ways in which entities (either parties or external entities) can affect the execution of protocols, we consider adversaries which can corrupt parties, forcing them to deviate from the protocol. In most works, there is a bound on the number of parties t adversaries may corrupt – in fact, in most models secure multiparty computation is only possible if no more than a minority of parties is corrupted, i.e. $t < n/2$). More general *adversarial structures* are possible [HM97].

The most general type of adversary is the one that may corrupt parties before and during the execution of the protocol, possibly choosing which parties to corrupt based on the data gathered so far. We say such an adversary is *adaptive*. A more restricted (but very common) model is one where the adversary must select the parties to corrupt before the protocol starts and the set of corrupted parties do not change during the execution. Such an adversary is called *non-adaptive* or *static*. An orthogonal aspect of adversarial capabilities is the degree in which the adversary can make parties to deviate from the protocol. If the corrupted parties can take active steps to disrupt the execution (say by sending messages not specified by the protocol or failing to reply) the adversary is said to be *active*. An adversary is *passive* (or *honest-but-curious*, *semi-honest*) if a dishonest party does not deviate from the protocol except to provide the adversary with all the information gathered so far. This last type of corruption is useful to model adversaries which may have access to private data from parties *after* the protocol has concluded. In some models, it may be useful to model parties which are allowed to recover after being corrupted (and thus became honest again), in which case the adversary may potentially corrupt all parties, as long as it does not corrupt more than a given bound during a "time period". This adversary has been called *mobile* or *proactive proactive*.

As mentioned before, the resources of the adversaries are usually considered polynomially bounded as it naturally captures realistic threats. In some settings, however, it is possible to prove meaningful results even against computationally unbounded

adversaries. Such results often assume private channels.[2] This model is called *information-theoretic* while the former is referred as the *computational* model.

Communication Channels: The adversary is commonly assumed to monitor all communication channels between honest parties. By monitor, we mean the adversary is able to obtain a copy of each message sent by each channel. A variant of this assumption is that the adversary can only monitor a subset of all the communication channels. (This is arguably realistic in most practical networks.) Alternatively, the *private-channel* model postulates communication channels between parties are untappable by the adversary. Although much stronger, this assumption can be justified in some settings, and it can be implemented (emulated) by tappable channels in some others (by using encryption). It can also provide a clean, simple abstraction to design secure protocols. Another standard assumption is that the adversary cannot modify, delete, duplicate, or create new messages for communication channels between honest parties. As with the previous assumption, in some settings this assumption can also be emulated using appropriate cryptographic tools like encryption and signatures (see for example [BR93, CK02]).

Network Topology: A standard assumption is that parties are pairwise connected with dedicated (two-way) communication channels, also known as *point-to-point channels*. Less standard is that some topology exists and communication is limited to a particular *graph* where parties are the nodes and communication channels are the edges. A stronger, but common, assumption is that parties share a *broadcast channel*, which allows some or all parties to sends a message to all the other parties so all parties are guaranteed to receive the same message (see details in Section 4.2.2). As before, this assumption can be justified in some settings: under some conditions it can be emulated by point-to-point channels (see [FM97] or [LLR02a] and references therein).

Synchronous versus asynchronous networks: Messages sent in communication channels are often assumed to be *reliably delivered* – they might be delayed but always arrive to their destination. In terms of delays, there are two standard assumptions: synchronous networks and asynchronous networks. In a *synchronous* network, parties

[2] If the definition of security follows the simulation paradigm, it is also required that the simulator (or ideal adversary) operates in time comparable with that of the adversary [Can00a]. Such results then imply results for computationally bounded adversaries.

have perfectly synchronized clocks which "tick" at discrete instants. The time interval between the i-th tick and the $(i+1)$-th tick is called the i-th round. Messages sent in one round are guaranteed to be delivered in the next round. A network is said to be *partially* synchronous if the adversary is allowed *rushing*, which means that the network delivers the messages addressed to corrupted players instantly, so the adversary obtains those messages before deciding and sending out the messages of corrupted players for the same round.[3] We say the communication channel is *asynchronous* if an attacker can succeed in delaying messages by an arbitrary, but finite amount of time. In other words, if any message inserted into a channel is eventually delivered.

SETUP ASSUMPTIONS: In some cases it may be helpful to assume parties do share or know some information *before* the protocol starts. For example, common assumptions are that parties know public keys (or verification keys) corresponding to each of the other parties, or that parties have access to a common (trusted) random string. The former assumption is called the *bare public key* (BPK) model, while the latter the *common-random string* (CRS) model.

A slight strengthening of the BPK model is the *public key infrastructure* model where the secret keys corresponding to public (verification) keys are securely associated to parties, i.e. parties are assumed to know their secret keys (see [Rey01] for more details). More precisely, in the PKI model, we assume all parties $P_1, \ldots, P_n$ hold the same vector $pk_1, \ldots, pk_n$ of public keys for a certain encryption scheme, and each party P_i holds a secret key sk_i corresponding to pk_i. Also, each key pair (pk_i, sk_i) is correctly generated for each (honest) party P_i.

The PKI model is assumed in several parts of this thesis.

In this work, we use several network models. In Section 3.2, we consider mostly the one with partially synchronous point-to-point channels plus broadcast channel, static, passive adversary (with no corruptions[4]). In Section 2.1.2, we strengthen the model to include adaptive, active adversaries that corrupt up to a minority $t < n/2$ of the parties.

[3] In some works in the literature, the partial synchronicity assumption is left implicit ("synchronous" meaning "partially synchronous") or the network is referred as "synchronous with rushing adversaries".

[4] In Section 3.6 we comment on how to extend our results for the case of corruptions, i.e. passive adversary corrupting up to t parties.

Finally, in Section 5.2, we use a generalization of the asynchronous model, the Universally Composable framework [Can05], which is described in Section 2.2.4.

2.2.2 Multiparty Setting: Building Blocks

In this section, we first review the concept of *verifiable secret sharing* [Sha79, Bla79, CGMA85], a very powerful and useful protocol which is extensively used as primitive in the design of most multiparty protocols. Then, we recall the definition of *key private asymmetric encryption schemes* [BBDP01] which are a key ingredient in the design of anonymous protocols of Chapter 3.

Verifiable Secret Sharing

A secret sharing scheme is a multiparty protocol where a certain party, the *dealer*, holds a secret s that wishes to distribute or *share* among n other parties in such a way that secret s can be reconstructed by any group of at least t parties. Also, it is often required that any subset of less than $t < n$ parties should not obtain any information on the secret, even after the secret has been shared. Secret sharing protocols were invented independently by Shamir [Sha79] and Blakley [Bla79].

Shamir Secret Sharing: Shamir's secret sharing [Sha79] is based on the idea of polynomial interpolation. Let F be any finite field, and $s \in F$. Associated to each party P_i, there is a point $x_i \in F$, where all points x_i are distinct (which requires $|F| \geq n$). The dealer chooses a polynomial $f(x) = a_0 + a_1 x + \ldots + a_{t-1}x^{t-1}$ by selecting coefficients $a_1, \ldots, a_{t-1} \in F$ uniformly at random. To share secret s, the dealer sets $s_0 = s$ and evaluates the polynomial f on the n different points $x_1, \ldots, x_n$. Then, the dealer privately sends $(x_i, f(x_i))$ to each party P_i. By the interpolation property of polynomials, t shares suffice to reconstruct f and hence s. Any set of less than t parties knows at most $t-1$ shares (as the shares are sent privately to each party by the dealer) which does not reveal any information about $s_0 = s$ as any $s' \in F$ defines a possible polynomial that satisfies these $t-1$ constraints and $f(0) = s'$.

There are two drawbacks in Shamir's scheme. A misbehaving dealer can deal

inconsistent shares to the participants (shares that are not points computed as instructed by the protocol), effectively preventing parties to reconstruct the secret. Also, dishonest parties at reconstruction stage may provide other parties with shares other than the ones received causing an incorrect secret to be reconstructed. Both problems are solved by *verifiable secret sharing* schemes, which were proposed by Chor, Goldwasser, Micali and Awerbuch [CGMA85]. In these schemes, each party can verify that the data received is indeed a correct share, and incorrect shares submitted by parties at reconstruction time can be detected.

CLASSICAL DEFINITION: We revisit the definition of verifiable secret sharing by Feldman and Micali [FM88, FM97] (cf. [GRR98]). The definition assumes a synchronous network with rushing adversaries, and relies on the notion of a *fixed event*: an event $\mathcal{X}$ is *fixed* at a given round r in an execution E of a protocol, if $\mathcal{X}$ occurs in any execution E' of the protocol coinciding with E up to round r.

Definition 2.2.1 A certain party D (say $D = P_1$) is referred to as the dealer; let K be the set of possible secrets. Consider a protocol π that consist of two phases: a *sharing phase* and a *reconstruction phase*.

(1) **Sharing:** The dealer holds an input $s \in K$, referred to as the *secret*, and each party P_i holds an independent random input r_i. At the end of this phase, each party P_i locally outputs its view v_i of this phase. This view includes a boolean value ver_i.

(2) **Reconstruction:** In this phase, each party holds as input its view v_i from the sharing phase. At the end of this phase, each player locally output an output value $y_i \in K$.

A two-phase, n-party protocol as above is called a (n, t)-VSS protocol if, for any adversary corrupting up to t parties, the following requirements hold:

- **Unanimity:** If any honest party P_i outputs $ver_i = 1$ at the end of the sharing phase, then $ver_j = 1$ for all the other honest parties P_j.
- **Acceptance of good secrets:** If D is honest, then $ver_i = 1$ for every honest party P_i.

- **Verifiability:** If an honest party P_i outputs $ver_i = 1$ at the end of the sharing phase, there exist a value $s^* \in K$ such that the event that all honest parties output s^* at the end of the reconstruction phase is fixed at the end of the sharing phase. Moreover, if D is honest, then $s^* = s$ the original input of D.
- **Privacy:** If D is honest, the adversary's view during the sharing phase is (perfectly) simulatable by a simulator T that only takes the description of K as input.

∎

In Section 2.2.4, we revisit the definition of verifiable secret sharing and show how it can be adapted for the universally composable security framework [Can05].

Key Private Encryption

KEY-PRIVATE ASYMMETRIC ENCRYPTION SCHEMES: The concept of *key privacy* for asymmetric encryption schemes was originally proposed by Bellare, Boldyreva, Desai and Pointcheval [BBDP01]. Motivated by anonymity-preserving applications, key privacy aims to guarantee that no useful information about the public key holder's identity should be obtained from a ciphertext encrypted under that public key, even if all public keys are known.

Let $\mathcal{AS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme.[5] Let $b \in \{0,1\}$, $k \in \mathbb{N}$ be the security parameter. Consider the following experiment:

Experiment $\mathbf{Exp}^{ik-cpa-b}_{\mathcal{AS},A}(k)$
$(pk_0, sk_0) \xleftarrow{R} \mathcal{K}(1^k)$, $(pk_1, sk_1) \xleftarrow{R} \mathcal{K}(1^k)$
$(x, s) \leftarrow A(\text{“find”}, pk_0, pk_1)$
$y \leftarrow \mathcal{E}_{pk_b}(x)$
$g \leftarrow A(\text{“guess”}, y, s)$
return g

[5] Our presentation differs from the one in [BBDP01] in that we do not consider a *common-key generation algorithm* which there is used to generate the public parameters common to all parties. Instead, for simplicity, we assume all common parameters for the encryption scheme are generated initially once and for all.

An encryption scheme $\mathcal{AS}$ achieves key privacy against chosen plaintext attack (IK-CPA) if the quantity

$$\mathbf{Adv}^{ik-cpa}_{\mathcal{AS},A}(k) \stackrel{\text{def}}{=} \Pr\left[\mathbf{Exp}^{ik-cpa-1}_{\mathcal{AS},A}(k)=1\right] - \Pr\left[\mathbf{Exp}^{ik-cpa-0}_{\mathcal{AS},A}(k)=1\right]$$

is negligible in k for any feasible (PPT in k) adversary A.

2.2.3 Standalone Security

One possible way to capture the security requirements of multiparty protocols is by seen them as a distributed algorithm to compute a certain *function*, where the function is computed on inputs provided by some of the parties (including those carrying out the computation). This approach is referred as the *secure function evaluation* model.

We briefly recall the definition of secure function evaluation from [Can00a]. Let $n \in \mathbf{N}$ be a fixed parameter. Assume Π is an n-party protocol, $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of inputs for the parties, and $f\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be a function that map n strings into n strings. Intuitively, the goal of protocol Π is to compute function f on the inputs held by the parties, in such a way that each party receives some output as specified by f. In order to define the security of a protocol computing f, the simulation paradigm is used and two processes or "worlds" are considered: an ideal process and a real process. In both, all communication is partially synchronous.

In the ideal process, there is a trusted third party connected to each party by a point-to-point channel. The trusted party may compute f on the inputs provided by the parties and returns the output to the parties as specified by f. In this world, the protocol $\mathsf{Ideal}(f)$ that computes f on the parties' input vector $\mathbf{x}$ is very simple: all parties privately submit their inputs to the trusted party, which computes $(y_1, \ldots, y_n) = f(\mathbf{x})$ and returns each output y_i to party P_i. Adversaries in the ideal process cannot corrupt the trusted party but can corrupt an arbitrary subset B of the parties before the protocol starts. Upon corruption, party P_j gives her input x_j to the adversary and all her outgoing messages and output is controlled by the adversary. The communication channel between each honest party and the trusted third party cannot be eavesdropped by the adversary. This process models the ideal case in which an adversary cannot disrupt the computation other than replacing the inputs and outputs of corrupted parties, nor

obtain more information from the inputs and outputs of the honest parties other than what can be inferred from the output of the corrupt parties.

In the real process there is no trusted third party and parties are connected by pairwise point-to-point channels. Adversaries in the real process can also corrupt an arbitrary set B of parties; corruption occurs as in the ideal process. Parties execute protocol Π in this world.

For any security parameter $k \in \mathbf{N}$, any input $\mathbf{x} = (x_1, \ldots, x_n) \in (\{0,1\}^*)^n$ for the parties, and any real-process adversary A with auxiliary input $z \in \{0,1\}^*$, we let $\text{EXEC}_A^\Pi(k, \mathbf{x})$ denote the $(n+1)$-vector whose elements are the output of each party and the adversary after executing protocol Π under adversary A. That is,

$$\text{EXEC}_A^\Pi(k, z, \mathbf{x}) \stackrel{\text{def}}{=} \left(\text{OUTPUT}_A^\Pi(k, z), \text{OUTPUT}_{P_1}^\Pi(k, x_1), \ldots, \text{OUTPUT}_{P_n}^\Pi(k, x_n) \right) .$$

where $\text{OUTPUT}_C^\Sigma(k, x)$ denotes the local output of entity C (either a party or adversary) after executing protocol Σ on input the security parameter k and value x. Similarly, for the ideal process, given any ideal-process adversary S the execution of $\mathsf{Ideal}(f)$ under adversary S is defined as

$$\begin{aligned} &\text{EXEC}_S^{\mathsf{Ideal}(f)}(k, z, \mathbf{x}) \\ &\quad \stackrel{\text{def}}{=} \left(\text{OUTPUT}_S^{\mathsf{Ideal}(f)}(k, z), \text{OUTPUT}_{P_1}^{\mathsf{Ideal}(f)}(k, x_1), \ldots, \text{OUTPUT}_{P_n}^{\mathsf{Ideal}(f)}(k, x_n) \right) . \end{aligned}$$

Both quantities define ensembles in the straightforward way,

$$\begin{aligned} \text{EXEC}_A^\Pi &\stackrel{\text{def}}{=} \left\{ \text{EXEC}_A^\Pi(k, z, \mathbf{x}) \right\}_{k \in \mathbf{N}, z \in \{0,1\}^*, \mathbf{x} \in (\{0,1\}^*)^n} \\ \text{EXEC}_S^{\mathsf{Ideal}(f)} &\stackrel{\text{def}}{=} \left\{ \text{EXEC}_S^{\mathsf{Ideal}(f)}(k, z, \mathbf{x}) \right\}_{k \in \mathbf{N}, z \in \{0,1\}^*, \mathbf{x} \in (\{0,1\}^*)^n} \end{aligned}$$

A protocol Π securely implements function f if for any real-process PPT adversary A there exists an ideal-process PPT adversary S such that EXEC_A^Π is computationally close to $\text{EXEC}_S^{\mathsf{Ideal}(f)}$, denoted

$$\text{EXEC}_A^\Pi \stackrel{c}{\approx} \text{EXEC}_S^{\mathsf{Ideal}(f)}$$

Spelled out, protocol Π securely implements function f if for any real-process PPT adversary A there exists an ideal-process PPT S such that for all PPT distinguishers D,

for all constant $c > 0$ and for all sufficiently large k, all $z \in \{0,1\}^*$ and $\mathbf{x} \in (\{0,1\}^*)^n$,

$$\Big|\Pr\Big[\, D(1^k, z, \mathbf{x}, \textsc{Exec}_A^{\Pi}(k, z, \mathbf{x})) = 1 \,\Big] - \Pr\Big[\, D(1^k, z, \mathbf{x}, \textsc{Exec}_S^{\mathsf{Ideal}(f)}(k, z, \mathbf{x})) = 1 \,\Big]\Big| \;<\; k^{-c}$$

2.2.4 Universally Composable Security

In this section, we provide a brief and informal overview of the Universally Composable framework of Canetti as described in [Can05]. The Universally Composable (UC) framework is an extension of the standalone security model of the previous section. This framework allows defining the desired properties of cryptographic protocols in terms of tasks or *functionalities*. A functionality is a program executed by a "trusted party" which first obtains inputs directly from the parties, performs certain instructions on these inputs, and provides the parties with the appropriate outputs.[6] Such protocol is called the *ideal protocol* and the functionality used is the *ideal functionality*. Functionalities can be reactive, in the sense that they may receive inputs from the parties and the adversary, as well as as deliver outputs to the parties and the adversary, in the course of one ore more iterations. Informally, a protocol securely implements a given cryptographic task, if executing the protocol against a realistic (i.e. real-life) adversary "emulates" an *ideal process* in which the task is computed by the functionality directly interacting with the parties against a limited adversary (called the *ideal adversary*). The model assumes there exist a third party, the environment $\mathcal{Z}$, which interactively gives inputs to the parties and sees the parties' outputs, possibly at several times during the computation. The environment can also freely communicate with the adversary. The objective of the environment is to distinguish whether the observed execution corresponds to a "real-life" execution of the protocol or whether it corresponds to the ideal process with the ideal functionality is used. In this sense, the notion of *emulation* of UC security effectively is based on that no interactive distinguisher cannot tell the executions apart. More precisely, a protocol π is said to *UC-securely realize* functionality $\mathcal{F}$ if for any real-life adversary A attacking protocol π under the presence of the environment

[6] A functionality may also take inputs from or give outputs to the adversary to model the "allowed influence" of the adversary in the computation of the task, or the "allowed leakage of information" to the adversary, respectively.

$\mathcal{Z}$, there exists an ideal adversary S for the ideal process such that no environment $\mathcal{Z}$ can distinguish whether it is interacting with the real-life adversary A and protocol π, or it is interacting with the ideal adversary S and the functionality $\mathcal{F}$.[7] Intuitively, this means that, whatever adversary A can obtain from the execution of protocol π, it can be obtained by S from the (obviously secure) ideal functionality $\mathcal{F}$, regardless of what external protocols may run concurrently or what inputs the parties start from.

The framework also consider protocols that operate in the real process but with access to an existent functionality $\mathcal{F}$. One benefit of assuming the availability of a given functionality is that it facilitates modular protocol design: as functionalities can be implemented and proven secure independently from the protocols that use them. Formally, a protocol is said to run in the *$\mathcal{F}$-hybrid model*, if it runs in the real-life process but parties have access to copies of functionality $\mathcal{F}$ (i.e. parties can send inputs and get outputs from an unbounded number of copies of $\mathcal{F}$ as in the ideal model).

In the UC framework, the adversary gets to see, and schedule the delivery of all the messages exchanged between parties – in particular, the adversary can block messages. However, adversaries can not see the parties' inputs nor the inputs that parties give to (or receive from) the functionality. (We stress parties cannot communicate directly among them, they can only communicate via the adversary). This models that communication between parties happens through potentially untrusted communication channels, while communication with any functionality is private and thus "trusted". The environment, on the other hand, has access to the parties' inputs and it obtains the parties' outputs, but it cannot see the messages exchanged between them. This shows that the adversary and the environment capture slightly different concerns: the adversary represents active threats via corruptions or by leaking information via communication channels, while the environment represents "everything else", all external (possibly concurrently executing) protocols, including possible inputs provided by them and possible outputs leaked by the interaction with them. In any case, the UC framework allows the environment and adversary to freely communicate during the protocol execution, so this restriction does not limit the model.

[7] The ideal adversary S is sometimes called the *simulator*, as S must simulate the real-life adversary's actions by somehow using the ideal functionality, without having access to the information exchanged in the real protocol.

In any hybrid model (or the ideal process), the adversary can communicate directly with the functionality as long as the functionality's code allows it. One common provision is that the adversary A may corrupt parties in a dynamic fashion (choosing which party to corrupt as a function of what A sees during the execution) by sending a special message $(\texttt{Corrupt}, P)$ to the functionality, where P is the party to corrupt. The functionality's code then determines the subsequent actions, like sending party P's inputs and local state to the adversary, for example. As expected, the definition of security mentioned before is also presented for protocols in the hybrid models. A protocol π in the $\mathcal{F}$-hybrid model *emulates* protocol ϕ in the $\mathcal{G}$-hybrid model if for any adversary A attacking π in the $\mathcal{F}$-hybrid model and any environment $\mathcal{Z}$, there exists an adversary S attacking ϕ in the $\mathcal{G}$-hybrid model, such that $\mathcal{Z}$ have at most negligible probability of distinguishing whether it is interacting with A in the former model, or S in the latter.

Universal Composition: The security of UC secure protocols is maintained under general composition with an unbounded number of instances of arbitrary protocols running concurrently. This follows from the composition theorem, proven by Canetti in [Can01b, Can05]. The theorem essentially states the following. Given a protocol ρ that UC-securely implements a functionality $\mathcal{G}$ in some $\mathcal{H}$-hybrid model (or the real-life model), then any protocol π that UC-securely implements a functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model can be transformed into a protocol π^ρ in which all calls to the ideal functionality $\mathcal{G}$ are replaced with calls to protocol ρ, in such a way that protocols π^ρ in the $\mathcal{H}$-hybrid model emulates protocol π in the $\mathcal{G}$-hybrid model

Delayed Outputs: In [Can05], Canetti suggests a mechanism that allows parties to "ask for permission" from the adversary before sending a message to a party, possibly to model adversarially-controlled delays. A functionality $\mathcal{F}$ is said to give (private) "delayed output" m to party P if, before giving m to P, $\mathcal{F}$ sends a message of the form $(\texttt{AskPermission}, sid, P)$ to the adversary, and only upon receiving a message of the form $(\texttt{PermissionGranted}, sid, P)$ functionality $\mathcal{F}$ gives m to P. This mechanism is explicitly used in the construction of Section 5.6.

Modeling Synchronous Communication: In [Can05], Canetti suggests a technique to represent the important network model of synchronous communication (with guar-

anteed delivery). Canetti's idea is to *encapsulate* the synchronicity capabilities in a functionality, which can be accessed by the parties seeking for the synchronicity. The functionality, denoted $\mathcal{F}_{SYN}$, is described in Figure 2.1. We use this functionality in a crucial way in the construction and analysis of our Simultaneous Broadcast solution in Section 5.5.

UC Verifiable Secret Sharing

In this section, we revisit the definition of Verifiable Secret Sharing in the Universally Composable framework. This notion was first formalized by Canetti in [Can01a] (and later improved in [Can05]). Here we describe the variant proposed by Abe and Fehr in [AF04] which includes the concept of "spooling" the secret, a syntactic technique that allows the dealer to announce to the adversary – via a `Spool` message – that a new functionality is being called. The adversary is thus able to adaptively corrupt the dealer before the dealer commits to a value.[8] The corresponding UC functionality is $\mathcal{F}_{VSS}$ described in Figure 2.2.

Definition 2.2.2 We say a protocol π achieves UC-VSS if π UC-realizes functionality $\mathcal{F}_{VSS}$. ∎

In this formalization, the VSS scheme still consists of two phases, the sharing phase (steps $(1), (2)$) and the reconstruction phase (step (3)). Unanimity, verifiability, and acceptance of good secrets properties trivially follow from the functionality code (and the fact its execution is trusted). Privacy holds because step (3), the reconstruction, is not executed unless $t + 1$ parties – that is, at least one honest party – agree on opening the secret.

[8] A similar technique appears in the formalization in [Can05], albeit implicitly in the way the functionality reacts to the corruption messages from the adversary.

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

$\mathcal{F}_{SYN}$ expects its SID to be of the form $sid = (sid', \mathcal{P})$, where $\mathcal{P}$ is a list of parties among which synchronization is to be performed. It proceeds as follows.

(1) At the first activation, initialize a round counter $r \leftarrow 1$.

(2) Upon receiving input $(\texttt{Send}, sid, M)$ from party $P \in \mathcal{P}$, where $M = \{(m_i, R_i)\}$ is a set of pair of messages m_i and recipient identities $R_i \in \mathcal{P}$, record (P, M, r) and output (sid, P, M, r) to the adversary. (If P later becomes corrupted then the record (P, M, r) is deleted.)

(3) Upon receiving a message $(\texttt{Advance-Round}, sid, N)$ from the adversary, do: If there exist uncorrupted parties $P \in \mathcal{P}$ for which no record (P, M, r) exists then ignore the message. Else:

1. Interpret N as the list of messages sent by corrupted parties in this round. That is, $N = \{(S_i, R_i, m_i)\}$ where each $S_i, R_i \in \mathcal{P}$, m_i is a message, and S_i is corrupted. (S_i is taken as the sender of message m_i and R_i is the receiver.)
2. Prepare for each party $P \in \mathcal{P}$ the list L_P^r of messages that were sent to it in round r by all parties in $\mathcal{P}$, both corrupted and uncorrupted.
3. Increment the round number: $r \leftarrow r + 1$.

(4) Upon receiving input $(\texttt{Receive}, sid)$ from a party $P \in \mathcal{P}$, output $(\texttt{Received}, sid, r, L_P^{r-1})$ to P. (Let $L_P^0 = \perp$.)

Figure 2.1 The synchronous communication functionality, $\mathcal{F}_{SYN}$ [Can05].

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

$\mathcal{F}_{VSS}$ expects its SID to be of the form $sid = (sid', D, \mathcal{P})$, where $\mathcal{P}$ is a list of parties among which sharing is to be performed. It proceeds as follows.

(1) Upon receiving input $(\texttt{Spool}, sid, v)$ from party $D \in \mathcal{P}$, set $s \leftarrow v$ and send $(\texttt{Spooled}, sid, D)$ to adversary S.

(2) Upon receiving input $(\texttt{Share}, sid, v')$ from party $D \in \mathcal{P}$, do

- If D is corrupted, set $s \leftarrow v'$.
- Send $(\texttt{Shared}, sid)$ to each party in $\mathcal{P}$ and the adversary S.

(3) Upon receiving input $(\texttt{Open}, sid)$ from $t+1$ distinct parties, send $(\texttt{Opened}, sid, s)$ to all parties in $\mathcal{P}$ and the adversary S.

Figure 2.2 The Verifiable Secret Sharing (with Spooling) functionality, $\mathcal{F}_{VSS}$.

3

Anonymous Communication

3.1 Introduction

Anonymous channels allow users to send and receive messages without revealing their identities. There are many applications for such channels, from protecting "whistle blowers" or guaranteeing source confidentiality in crime tips, to offering access to medical information to potential patients without fear of embarrassment, or protecting voter privacy in electronic voting [FOO92, Nef01]. Chaum [Cha81] initiated the modern study of anonymous communication by introducing the concept of mix networks (or *mix-nets*). A mix-net is a protocol in which messages (say, emails) traverse several routers (or mixers) and, in the process, are "mixed" with other messages with the intention that the relation to the original sender be lost. Since Chaum's seminal paper, research in the area has been extensive, from concrete mix-net proposals (see [PPW91, Abe98, JJR02, FS01, Gro03, Wik04] among many others) to very practical protocols based on mix-nets (eg. [GRS96, GT96, KEB98, DDM03, RP04, DMS04] and references therein). But mix-nets are not the only method to implement anonymous communication. DC-nets (also known as anonymous broadcast networks), also proposed also by Chaum [Cha88] and later improved by many others [BdB89, Wai90, WP89, GJ04], allow broadcast of messages without disclosing the sender identity. At least initially, most of the effort was put into improving the efficiency and reliability of the constructions, so informal or ad-hoc definitions were common. Indeed, only recently the need for

general (and sound) definitions for these types of primitives has drawn some attention. Furukawa [Fur04] and Nguyen et al. [NSNK04], in particular, give strong definitions for "proving shuffles" (shuffles are the basic mixing operation) and Wikström [Wik04] presents a formal definition of mix-net in the UC model [Can01b]. These definitions, although helpful in the design and analysis of mix-nets, do not provide a definition of anonymous channels per se. Indeed, the absence of good anonymity definitions that capture realistic concerns motivated this work.

Our Contributions: We present a novel framework to organize and compare anonymity definitions. In this framework, we formalize the notions of unlinkability, sender-anonymity, receiver-anonymity, sender-receiver anonymity, and unobservability, giving them new, strong indistinguishability-based formulations without compromising the standard "intuitive" meaning they have in the literature [PK01]. We also introduce new notions, namely weak unlinkability, sender unlinkability and receiver unlinkability. These notions, while arguably weak, can be used to implement some of the stronger notions. Then we formally prove some folklore results: we show that sender-receiver anonymity implies both sender anonymity and receiver anonymity, that sender-anonymity and receiver-anonymity (both separately) imply unlinkability, and that unobservability implies all the other properties. In the other direction, we present generic black-box transformations from any "weak" anonymous protocols (eg. weak unlinkability, unlinkability, or sender anonymity) into protocols anonymous under "stronger" notions (like sender-receiver anonymity or unobservability). These transformations are provably optimal in terms of message traffic. We then revisit the anonymity of constructions based on broadcast channels, DC-nets and mix-networks, giving an exact characterization of the anonymity they provide in our framework.

3.1.1 Coping with information leaks

There have been several attempts to characterize the intuitive properties anonymous channels should have. Most proposals so far seem to fall into two categories: (a) they present intuitive but weak definitions (targeted to particular applications with efficiency in mind), or (b) they present strong definitions with often impractical imple-

mentations [BGW88, GMW86, CCD88]. We seek to bridge this gap by providing strong definitions which can be tailored to specific practical scenarios.

We identify factors or conditions that may realistically *limit* anonymity. These conditions are on specific information that, in principle, may be unrealistic to assume hidden from the adversary. Consider for example,

(a) **Total network flow is usually public:** the total number of messages sent in a system is likely to be known to any party in the system, even external observers.

(b) **Amount of traffic per party is hard to conceal:** the number of messages sent or received by a particular party is often easily inferred by an observer in the party's network vicinity.

(c) **Values sent or received by each party are not necessarily private:** the value of each message[1] sent or received by a particular party could be guessed, known, or even influenced by an adversary.

A proper definition of anonymity should take these "leaks" into account but hide any additional information: *hide everything except what follows from the potentially leaked information*. This idea is already present in security definitions of other cryptographic primitives. For example, if E is a semantically secure encryption function [GM84], it is standard to assume a ciphertext $E(m)$ hides all partial information about a message m except its length $|m|$. This is because $|m|$ can only be hidden at the cost of unnecessarily increasing the size of $E(m)$. In fact, the definitions in this work are inspired by the indistinguishability-based formalization of semantically secure encryption in [GM84], which guarantees the hiding of all information on the plaintext other than the plaintext length. Similarly, an anonymous channel should hide all information about the communication except for (some of) the information mentioned above. In this work, we study the possible combinations of the conditions (a),(b), and (c) above, and analyze the resulting notions. There are ten (potentially different) notions. Named following the intuition in [PK01], they are summarized in Table 3.1.

[1] We distinguish two properties for each message: its value, that is, the data or *payload* encoded in the message, and its destination.

Table 3.1 Anonymity variants and their alternative names. The alternative name (X, Y, Z) encodes what information is not assumed to be protected by the definition (ie. the meaning of X,Y, and Z), and from whom the information comes: from each sender (X), each receiver (Y), or both (Z). 'U' stands for "`values of the messages sent/received`", 'Σ' for "`number of messages sent/received`", '#' for "`total number of messages`", and '?' for "`nothing`".

Anonymity Variant	Alternative Name
Weak Unlinkability	$(\mathsf{U}, \mathsf{U}, \cdot)$-anonymity
Sender Unlinkability	$(\Sigma, \mathsf{U}, \cdot)$-anonymity
Receiver Unlinkability	$(\mathsf{U}, \Sigma, \cdot)$-anonymity
Sender-Receiver Unlinkability	$(\Sigma, \Sigma, \cdot)$-anonymity
Sender Anonymity	$(?, \mathsf{U}, \cdot)$-anonymity
Receiver Anonymity	$(\mathsf{U}, ?, \cdot)$-anonymity
Strong Sender Anonymity	$(?, \Sigma, \cdot)$-anonymity
Strong Receiver Anonymity	$(\Sigma, ?, \cdot)$-anonymity
Sender and Receiver Anonymity	$(?, ?, \#)$-anonymity
Unobservability	$(?, ?, \cdot)$-anonymity

Toy examples of the traffic patters protected by all variants are shown in Section 3.1.2. *Weak Unlinkability* is the weakest notion of anonymity we consider. A protocol is weakly unlinkable if it hides any relation between senders and receivers beyond what is implied by the specific values of the messages exchanged. Compared to Weak Unlinkability, *Sender Unlinkability* strengthens the requirements for the sender, hiding the value of the messages sent by a party, but not necessarily their total size. Its dual notion is *Receiver Unlinkability* in which the roles of sender and receiver are reversed. Next notion is *Sender and Receiver Unlinkability* (or simply *Unlinkability*), in which protocols do protect message values, but it may not protect the total size of messages exchanged by each party. A stronger notion is *Sender Anonymity* as the number and values of messages for the sender must remain hidden (but not the values of the received messages for each party). Compared to Sender Anonymity, *Receiver Anonymity* simply reverses the roles of sender and receiver. Further strengthening of these notions are *Strong Sender Anonymity* (resp. *Strong Receiver Anonymity*) in that protocols can afford to leak at most the amount of traffic per receiver (resp. per sender). The strongest notions are *Sender-Receiver Anonymity*, and *Unobservability*. They differ in that the former may not protect the total network flow (ie. the total number of messages exchanged), while the latter must hide this information.

3.1.2 Strong, Formal Definitions

We adopt an indistinguishability based formalization under which the adversary produces two message matrices (which encode message senders and receivers in a standard way), is allowed to passively observe the execution of a communication protocol under a random one of these two matrices and then is required to have non-negligible advantage in determining under which of the two matrices the protocol was executed. Within this framework, each different anonymity variant is defined by requiring the adversary to produce two matrices whose "leaked" information is the same. More precisely, if for any message matrix M the anonymity variant assumes a certain information $f(M)$ may not be protected (it may be "leaked"), then the two matrices M, M' produced by the adversary must satisfy $f(M) = f(M')$. Indeed, the notions corresponding to the different anonymity variants mentioned in the previous section follow from instantiating function f with the appropriate function (eg. one that computes the set of message values sent per party, their number, or the total number of messages, for example). Our formalisms build on definitional ideas used for encryption [GM84, MRS88, Gol93] and signatures [GMR88]. Regarding adversaries, an often adopted adversarial type is that of *honest-but-curious* (or passive) adversary, one where the adversary obtains the internal state of the corrupted party, but the party continues to follow the protocol. For simplicity of exposition, we consider passive adversaries with no corruptions (also called *outside* [DO00] or *global passive adversary* [Ser04]) as it captures most of the subtleties of our model. Extensions to allow (passive) corruptions are discussed in Section 3.6.

Since the adversary can freely choose the values and destinations of all messages in the protocol (ie. the message matrix), it follows that a protocol anonymous under this definition must hide all partial information on the message matrix M *except for what is implied by the known information* $f(M)$. In particular, sources and destinations of the messages are hidden up to the extent that they do not follow from the known information. This is quite a strong guarantee.

We stress that we present an unified framework for *all the proposed anonymity variants*. We believe this facilitates the organization and comparison of the notions as well as future extensions.

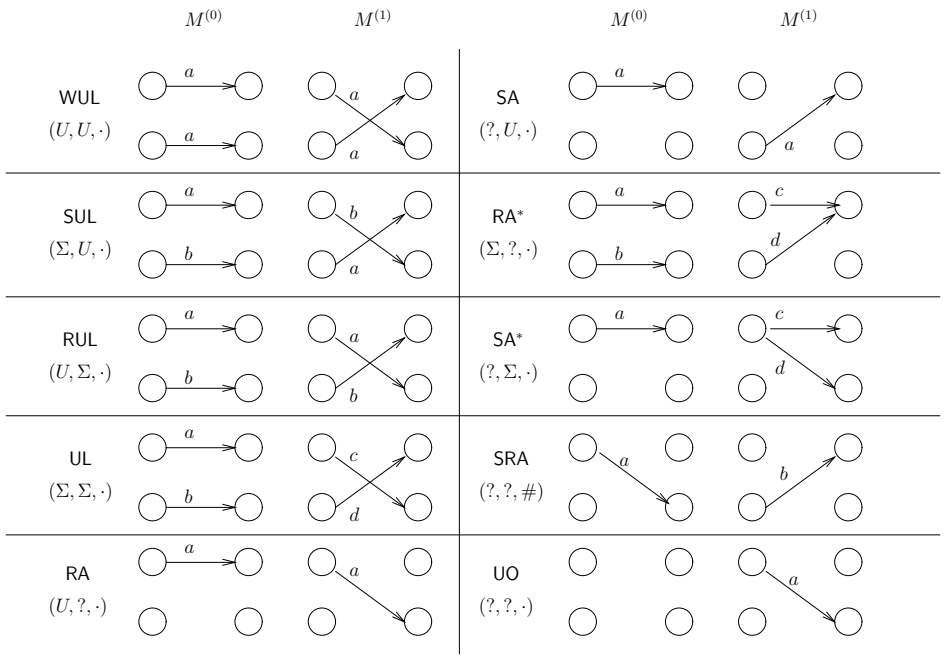

Figure 3.1 A pictorial representation of toy examples of communication patterns hidden by each anonymity notion. For each notion, there are two communication patterns illustrated by graphs of four nodes: the leftmost graph represents the communication pattern for the combination of senders, messages, and receivers corresponding to matrix $M^{(0)}$, while the rightmost graph the pattern specified by $M^{(1)}$. For each graph the nodes which represent parties, arrows represent messages, and the label is the message value; the nodes where arrows depart represent senders, and those where arrows arrive represent receivers.

3.1.3 Comparing Notions

The indistinguishability-based definitions presented in this chapter appear to capture the concerns of most intuitive but informal notions of anonymity proposed in the past [PK01]. Indeed, in Section 3.1.5 we argue that previous anonymity formalizations in comparable network models are implied by some of the proposed notions. In addition, we compare the new notions to each other. The comparison is in terms of reductions. We distinguish two kind of them: notion A implies (is stronger than) notion B if (1) any protocol satisfying A can be used to achieve B (via a different protocol), or (2) any protocol satisfying A also satisfies B. A difficulty arises if we assume point-to-point channels between parties. In this case, protocols for all notions exist because of general

secure multiparty computation results [BGW88, GMW86, CCD88], which makes the notions trivially equivalent. To avoid this pitfall, we assume that the only communication channel between the parties is an idealized version of a protocol achieving notion A, and then we show how to implement a protocol that achieves notion B in this setting. The communication channel is idealized in the sense that parties only see its input/output behavior. This effectively gives us black-box reductions.

Results: We show three types of reductions between the anonymity definitions: (1) Trivial reductions, in which given a protocol for notion A, the same protocol achieves notion B, (2) Reductions that use cryptography, and (3) Reductions that use "padding" (or "dummy traffic"). Interestingly, in terms of the reductions, cryptography and padding do not appear exchangeable. Our results suggest that in the reductions that require cryptography padding does not help, while in those where padding is necessary, cryptography does not help.

Trivial Reductions: There exists a partial order of the notions, starting from the weakest one, weak unlinkability, and ending in the strongest one, unobservability, such that if a protocol achieves a certain notion then the same protocol achieves any weaker notion. These relations give formal justification to previous informal statements such as sender-receiver anonymity implying both sender anonymity and receiver anonymity, or that unobservability implies all the other notions. Interestingly, there is no trivial relation between sender anonymity, unlinkability, and receiver anonymity, which indicates the definitions address incomparable security concerns. In [PK01], however, it is argued that Unlinkability (called "relationship anonymity" there) is a "weaker property than each of sender anonymity and recipient anonymity". The disagreement disappears when one notices that, under our definitions, such relation is true between *strong* sender (or receiver) anonymity and unlinkability. Our framework allows us then to clarify an implicit assumption in [PK01], namely that messages in the definitions of sender and receiver anonymity are private.

Using Cryptography: Under standard computational and setup assumptions, we show that anonymity notions that reveal message values are not intrinsically weaker than those that keep these values private. In particular, we show reductions from unlinkability

to sender (or receiver) unlinkability and, perhaps more surprisingly, from any of them to weak unlinkability. We also show strong sender (resp. receiver) anonymity is not weaker than sender (resp. receiver) anonymity.[2] The assumptions are standard, namely PKI and key-private secure encryption schemes [BBDP01].[3] The reductions are computationally efficient and do not have message overhead – they introduce no new messages – therefore optimal in terms of communication.

Using "Padding": We conclude showing that our strongest anonymity notions *can* be achieved starting from much weaker anonymity notions, but at a cost of message efficiency. In a nutshell, the reductions show that unobservability, sender-receiver anonymity, strong sender (or receiver) anonymity, and unlinkability are actually equivalent. They also show that neither sender nor receiver unlinkability are stronger than sender or receiver anonymity. These reductions do introduce *dummy traffic* (ie. extra empty messages) but no more than necessary – they have optimal message overhead. These reductions do not require computational or setup assumptions, and are computationally efficient.[4] The results are summarized in Figure 3.3.

3.1.4 The Anonymity of Previous Protocols

The ultimate purpose of a definition is to be used to properly characterize the security of concrete protocols. Accordingly, we revisit the security of known constructions based on broadcast channels [BIK+03], DC-nets or anonymous networks [Cha88, GJ04, SA00], and mix-nets [Gro03, NSNK04, Fur04]. In Section 3.5, we examine the basic construction of Blaze et al. [BIK+03], which is based on broadcast channels, and we argue it can be shown *strong receiver anonymous*. We also discuss the DC-nets of [GJ04] and sketch how the construction there can be proven *sender anonymous*. Finally, we highlight sufficient conditions to prove the *strong receiver anonymity* of mix-net constructions based on shuffles [Gro03, NSNK04]. By combining the constructions that underlie the

[2] This proof actually *justifies* the assumption made in [PK01] mentioned before. We stress that this is not obvious since anonymity does not necessarily implies message privacy, or viceversa.

[3] In fact, based on preliminary results, we conjecture computational or setup assumptions are also necessary.

[4] The reductions *to* Sender Anonymity, Strong Sender Anonymity, and Unobservability require the extra (but rather mild) assumption that a known upper bound on the total network flow exists. See Proposition 3.4.6 and remarks at the end of Section 3.4.2.

implications of previous sections, we obtain anonymous protocols provably secure under the strongest notions: *sender-receiver anonymity* and *unobservability*.

3.1.5 Comparison with Previous Anonymity Notions

In this section, we compare the proposed variants with anonymity variants suggested previously in the literature. When necessary, we relax those definitions to match our adversarial model (passive adversaries with no corruptions).

INDISTINGUISHABILITY-BASED DEFINITIONS: Beimel and Dolev [BD03] define anonymity in terms of computational indistinguishability of the adversary's *view* (i.e. the messages and any extra information obtained by the adversary) in two cases: when party P_i sends a message to party P_j, and when $P_{i'}$ sends a message to $P_{j'}$, for any i, j, i', j'. Given that [BD03] does present protocols for multiple senders, we see the definition as somewhat unsatisfactory in the following sense. The definition does not specify how the messages and destinations for parties $P_k \neq P_i$ are selected. If they are chosen either arbitrarily (but the same for both views) or with some probability distribution, then we can show they are strictly *weaker* than sender-receiver anonymity. The alternative, choosing the inputs for parties $P_k \neq P_i$, arbitrarily but different in each view, might work (be equivalent to sender-receiver anonymity) although it is unclear without a formal statement. A similar concern can be raised on the definition proposed by von Ahn et al. in the context of k-anonymity [vABH03]. (Essentially the same definition for the case of a fixed receiver).

Golle and Juels [GJ04] present a definition of anonymity (which they called privacy) in the context of DC-nets [Cha88]. In the definition in [GJ04], a successful adversary must distinguish between an execution where P_1 sends a message to some party P_b, and one in which P_2 sends a message to some party P_{1-b}, where b is a bit chosen uniformly at random and *unknown* to the adversary. The rest of the parties sends messages as instructed by the adversary. Unfortunately, this definition suffers from a problem similar to the one above. The adversary is unable to exploit possible correlations between the destination of P_1's message and the destination of some other party P_3's message. Consequently, this definition can be shown to be strictly weaker than our definition of sender anonymity. Luckily, the DC-net in [GJ04] is strong enough

to be proven sender anonymous (see Section 3.5.2).

OTHER CLOSELY RELATED DEFINITIONS: Nguyen et al. [NSNK04] define privacy of a shuffle by a similar experiment to ours (a notion called indistinguishability under chosen permutation attack or IND-CPA$_S$ under an active adversary). In their definition, the adversary chooses two permutations under which the messages are shuffled and must distinguish which one was used. Translated to our setting, their definition restricts message matrices to be permutations such that each party sends exactly a single message. Also, it does not account for the types of information leaks we consider. The comparison is somewhat unfair, as their concern – privacy of a single shuffle – is different than ours.

Recently, another related definition was suggested (rather implicitly) by Ishai et al. in [IKOS06]. There, Ishai et al. describe a functionality for anonymous communication (synchronous setting with rushing). When paired with the appropriate notions of multiparty computation [Can00b] (under our adversarial model), their definition becomes a special case of ours, without the relaxations for the information leaks. Their work [IKOS06], however, does not explore the proposed definition but instead use it to prove the security of other (non-anonymity related) cryptographic protocols. None of the definitions above incorporates provisions to deal with "leaked" information though.

3.1.6 Discussion and Related work

Dolev and Ostrovsky [DO00] present "xor-trees" protocols, a generalization of DC-net into a spanning tree, which they prove secure under a notion based on the concept of anonymity set (see below). Similarly, von Ahn et al. [vABH03] propose the notion of k-anonymity which can be seen as an extension of the DC-net model to more practical graph structures (which partition the parties into k-sized autonomous groups). Another approach was proposed by Rackoff and Simon in [RS93]. They describe a protocol for anonymous communication based on sorting networks, which is shown to satisfy some statistical mixing properties. Relaxations to weaker adversaries were proposed by Reiter and Rubin [RR98] and Berman et al. [BFTS04]. Both works presented alternative notions of anonymity as well as efficient constructions assuming an adversary that does not monitor all communication channels. Recently, Camenisch and Lysyanskaya [CL05]

give a formal definition of onion routing [GRS96] (along a provable secure protocol) but they explicitly avoid defining anonymous channels.

An alternative characterization of anonymity has been through the concept of anonymity set [Cha88, KEB98]. The anonymity set is defined as the set of parties that could have sent a particular message as seen from the adversary [PK01]. Follow up works [KEB98, SD02, DSCP02] have proposed new characterizations of anonymity, mostly in terms of the probability distributions the adversary assigns to each party in order to represent the likelihood such party is the sender of a message. Finally, definitions based on formal methods have also been proposed [SS99, HO04, Ser04, MVdV04, GHPvR05].

ORGANIZATION: The rest of the chapter is organized as follows. Section 3.2, introduces some notation and details on the execution model. Then, in Section 3.3, we present the formal definition of anonymous channels. Section 3.4 presents implications between the notions as well as proofs of their optimality in terms of communication. Then, in Section 3.5, we revisit previously proposed anonymous protocols and examine their security in the current framework. We conclude in Section 3.6 mentioning some extensions to the model.

3.2 Preliminaries

In this chapter, we consider a synchronous point-to-point network of n parties (see Section 2.2.1). We distinguish two (possibly overlapping) types of parties: senders and receivers. For any two finite sets A and B, let $A \uplus B$ denote the multiset union (also called sum or join) of A and B, and $|A|$ denote the size of multiset A. By convention, we assume the i,j-th element of any matrix $M = (m_{i,j})_{i,j\in[n]}$ are denoted by the matrix name in lowercase with subindexes i,j. As usual, M^T denotes the transpose of any matrix M, $m_{i,*} = (m_{i,j})_{j\in[n]}$ is a matrix row.

MESSAGES: We let $V = \{0,1\}^\ell$ denote the message space, where $\ell = \ell(k)$ and $k \in \mathbb{N}$ is the security parameter. The collection of messages sent by parties as well as their destinations is an $n \times n$ matrix $M = (m_{i,j})_{i,j\in[n]}$. For row index i and column index j,

$m_{i,j} \in \mathcal{P}(V)$ is the set of messages from party P_i to party P_j.[5] We call such a matrix $M \in \mathcal{M}_{n\times n}(\mathcal{P}(V))$ a *message matrix*. The *size* of matrix M, i.e. the total number of messages sent, is denoted by $|M| \stackrel{\text{def}}{=} \sum_{i,j\in[n]} |m_{i,j}|$.

Adversaries and Protocol Execution: In our setting, adversaries are (possibly external) PPT parties in the system which can passively monitor all the communication between parties. We consider only *passive adversaries*, stateful adversaries that do not corrupt any party but are able to read (but not alter) all the messages exchanged by the parties. A protocol π is a sequence of instructions that all parties (senders and receivers) must follow. The instructions involve local computations and point-to-point message exchanges between parties. Our execution model is a special case of the model presented by Canetti [Can00b] (since we consider only passive adversaries). Given a message matrix M, we define the execution of protocol π with input M under adversary A, as the real world process where each party P_i follows the instructions of protocol π using as input the i-th row $m_{i,*}$ of matrix M. In this process, we allow the adversary A to obtain a copy of all messages exchanged in all communication channels. We say protocol π is a *message-transmission protocol* if, for any PPT adversary A and any message matrix M, each receiver P_j's local output y_j after executing π on input M equals the multiset $\uplus_{j\in[n]} m_{i,j}$.

3.3 Security Notions

Our definition is formalized in an *indistinguishability-type experiment* following similar approaches used in the formalization of semantically secure encryption schemes [BDPR98]. We define anonymity via an *experiment* or *game*, in which there are two "worlds" (world 0 and world 1). We allow the adversary to choose the messages (values and destinations) sent by each party in each world. These choices are represented by two message matrices $M^{(0)}$ and $M^{(1)}$. Then, world $b \in \{0, 1\}$ is chosen uniformly at random, and message-transmission protocol π is executed by all parties on input $M^{(b)}$. We measure the adversary's success in terms of her ability to distinguish the two worlds.

[5] Actually, we abuse the notation and we see elements of $\mathcal{P}(V)$ as multisets. This extension is needed to consider parties that send duplicated messages to the same receiver (see Section 3.4.2).

Our definition is inspired by the standard game used to define semantically secure encryption scheme, namely the *left-or-right* characterization of IND-CPA [BDPR98]. There, the adversary arbitrarily chooses two messages of the same length, is returned an encryption of a random one of the two messages and then is required to guess under which message the encryption was generated. The adversary's inability to distinguish the plaintext underlying in the ciphertext effectively means she cannot compute any information on the plaintext except its length [GM84, BDPR98]. Similarly, the definition of our anonymity game guarantees that no information can be efficiently computed on the destinations of the messages sent during the protocol.

As mentioned in the introduction, one important difference between our formulation and the left-or-right game mentioned above is that we restrict the adversary's choices of the values and destinations of the messages to capture what is known to the adversary. These restrictions are captured as follows. Let f_{U}, f_{Σ}, and $f_{\#}$ be functions that maps matrices $M = (m_{i,j})_{i,j\in[n]}$ into $\mathcal{P}(V)^n$, $\mathbb{N}^n$, and $\mathbb{N}$ respectively, defined by $f_{\mathsf{U}}(M) \stackrel{\text{def}}{=} (\uplus_{j\in[n]} m_{i,j})_{i\in[n]}$, $f_{\Sigma}(M) \stackrel{\text{def}}{=} (\sum_{j\in[n]} |m_{i,j}|)_{i\in[n]}$, and $f_{\#}(M) \stackrel{\text{def}}{=} |M|$. Also, let $f_{\mathsf{U}}^T(M) \stackrel{\text{def}}{=} f_{\mathsf{U}}(M^T)$, and $f_{\Sigma}^T(M) \stackrel{\text{def}}{=} f_{\Sigma}(M^T)$. Associated to each function f there is a relation $R_f \subset \mathcal{M}_{n\times n}(\mathcal{P}(V))^2$ where $(M, M') \in R_f$ if and only if $f(M) = f(M')$. For simplicity, we denote $R_{\mathsf{U}} = R_{f_{\mathsf{U}}}$, $R_{\mathsf{U}}^T = R_{f_{\mathsf{U}}^T}$, $R_{\Sigma} = R_{f_{\Sigma}}$,$R_{\Sigma}^T = R_{f_{\Sigma}^T}$, and $R_{\#} = R_{f_{\#}}$.

We are now ready to present the main definition. Given an n-party message-transmission protocol π, an adversary A, and label $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{UL}, \mathsf{SA}, \mathsf{RA}, \mathsf{SA}^*, \mathsf{RA}^*, \mathsf{SRA}, \mathsf{UO}\}$, consider the experiment $\mathbf{Exp}_{\pi,A}^{\mathbf{N}-anon}(k)$ described below. Relation $R_{\mathbf{N}}$ is defined in terms of R_{U},R_{U}^T,R_{Σ},R_{Σ}^T and $R_{\#}$ according to the table in Figure 3.2. We define the success probability of adversary A attacking protocol π under notion $\mathbf{N}$ as $\mathbf{Adv}_{\pi,A}^{\mathbf{N}-anon}(k) \stackrel{\text{def}}{=} 2 \cdot \Pr\left[\mathbf{Exp}_{\pi,A}^{\mathbf{N}-anon}(k) = 1\right] - 1$ where the experiment is defined as follows:

Experiment $\mathbf{Exp}_{\pi,A}^{\mathbf{N}-anon}(k)$
  $b \stackrel{R}{\leftarrow} \{0,1\}$, and $\langle M^{(0)}, M^{(1)}\rangle \leftarrow A(k)$
  if $\langle M^{(0)}, M^{(1)}\rangle \notin R_{\mathbf{N}}$ **then return** 0
  else Execute π on input $M^{(b)}$ under adversary A until A outputs a bit g.
    if $(b = g)$ **return** 1 **else return** 0

$\mathbf{N}$	Notion	Description of $R_{\mathbf{N}}$
WUL	Weak Unlinkability	$R_{\mathsf{WUL}} \stackrel{\text{def}}{=} R_{\mathsf{U}} \cap R_{\mathsf{U}}^T$
SUL	Sender Unlinkability	$R_{\mathsf{SUL}} \stackrel{\text{def}}{=} R_{\Sigma} \cap R_{\mathsf{U}}^T$
RUL	Receiver Unlinkability	$R_{\mathsf{RUL}} \stackrel{\text{def}}{=} R_{\mathsf{U}} \cap R_{\Sigma}^T$
UL	Unlinkability	$R_{\mathsf{UL}} \stackrel{\text{def}}{=} R_{\Sigma} \cap R_{\Sigma}^T$
SA	Sender Anonymity	$R_{\mathsf{SA}} \stackrel{\text{def}}{=} R_{\mathsf{U}}$
RA	Receiver Anonymity	$R_{\mathsf{RA}} \stackrel{\text{def}}{=} R_{\mathsf{U}}^T$
SA*	Strong Sender Anonymity	$R_{\mathsf{RA}^*} \stackrel{\text{def}}{=} R_{\Sigma}$
RA*	Strong Receiver Anonymity	$R_{\mathsf{SA}^*} \stackrel{\text{def}}{=} R_{\Sigma}^T$
SRA	Sender-Receiver Anonymity	$R_{\mathsf{SRA}} \stackrel{\text{def}}{=} R_{\#}$
UO	Unobservability	$R_{\mathsf{UO}} \stackrel{\text{def}}{=} \mathcal{M}_{n\times n}(\mathcal{P}(V))^2$

Figure 3.2 Anonymity variants and their associated relations $R_{\mathbf{N}}$.

Definition 3.3.1 (Anonymous Channels) A message-transmission protocol π achieves $\mathbf{N}$-*anonymity* for $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{UL}, \mathsf{SA}, \mathsf{RA}, \mathsf{SA}^*, \mathsf{RA}^*, \mathsf{SRA}, \mathsf{UO}\}$, if for all PPT adversaries A, the quantity $\mathbf{Adv}_{\pi,A}^{\mathbf{N}-anon}(k)$ is negligible in $k \in \mathbb{N}$. ▌

3.4 Relation between the notions

Black-box implications: As mentioned before, we consider a simplified network where the only communication channel between the parties is an idealized implementation of a protocol satisfying a certain anonymity notion N_1. We say notion N_1 *implies* notion N_2 (or alternatively that N_2 *reduces to* N_1), denoted by $N_1 \rightarrow N_2$, if there exists a protocol $\theta^{(\cdot)}$ (with access to the idealized communication channel) such that, for all protocol π, the following holds: if π achieves N_1-anonymity, then θ^{π} achieves N_2-anonymity.

Results: Our results are summarized in Figure 3.3. We first describe some easy implications, most of them folklore results, which until now remained without formal proof. An interesting aspect of the result is that the transformation which enables the reductions is the identity function. Therefore, some definitions are stronger than others in the sense that any protocol achieving one definition also achieves the other one.

Proposition 3.4.1 The following implications hold unconditionally $\mathsf{UO} \rightarrow \mathsf{SRA} \rightarrow \mathsf{SA}^* \rightarrow \mathsf{SA} \rightarrow \mathsf{SUL} \rightarrow \mathsf{WUL}$, $\mathsf{SRA} \rightarrow \mathsf{RA}^* \rightarrow \mathsf{RA} \rightarrow \mathsf{RUL} \rightarrow \mathsf{WUL}$, $\mathsf{SA}^* \rightarrow \mathsf{UL} \rightarrow \mathsf{RUL}$ and $\mathsf{RA}^* \rightarrow \mathsf{UL} \rightarrow \mathsf{SUL}$. ▮

Proof of Proposition 3.4.1: First, we notice that, by definition $R_{\mathsf{U}} \subset R_{\Sigma} \subset R_{\#}$ and $R_{\mathsf{U}}^T \subset R_{\Sigma}^T \subset R_{\#}$. The results follows easily from these relations. We illustrate this by proving the implication $\mathsf{SUL} \rightarrow \mathsf{WUL}$. The other implications are similar. In order to prove that $\mathsf{SUL} \rightarrow \mathsf{WUL}$, it suffices to show that, for any protocol π, given a good WUL-adversary A, there exists a good SUL-adversary A'. Since $R_{\mathsf{U}} \subset R_{\Sigma}$, then it follows that $R_{\mathsf{WUL}} \subset R_{\mathsf{SUL}}$ and, in consequence, any WUL-adversary A for protocol π is also a SUL-adversary for the same protocol, so taking $A' = A$ suffices. ▮

3.4.1 Implications under computational assumptions

In this section, we show that, under some setup and computational assumptions (namely PKI and key-private secure encryption [GM84, BBDP01]), some of the notions are equivalent in the sense that a protocol achieving one definition can be efficiently transformed into a similar protocol achieving the other definition. In particular, WUL,RUL,SUL, and UL are all equivalent, as well as SA and SA^*, and RA and RA^*. The assumptions are formalized in Section 2.

Lemma 3.4.2 Assume key-private semantically secure public-key encryption schemes and PKI exist. Then $\mathsf{WUL} \rightarrow \mathsf{SUL} \rightarrow \mathsf{UL}$, $\mathsf{WUL} \rightarrow \mathsf{RUL} \rightarrow \mathsf{UL}$, $\mathsf{SA} \rightarrow \mathsf{SA}^*$ and $\mathsf{RA} \rightarrow \mathsf{RA}^*$. ▮

For each implication of the lemma, the structure of the proof is the same and is divided into two steps. To prove that notion $\mathbf{N}$ implies notion $\mathbf{N}'$, we first define an intermediate notion, called $\mathbf{I}$-$\mathbf{N}$-*anonymity* (or *value oblivious* $\mathbf{N}$-*anonymity*), which we prove is implied by $\mathbf{N}$, that is, $\mathbf{N} \rightarrow \mathbf{I}$-$\mathbf{N}$. Then, we prove that $\mathbf{I}$-$\mathbf{N} \rightarrow \mathbf{N}'$. Interestingly, the proof that $\mathbf{N} \rightarrow \mathbf{I}$-$\mathbf{N}$ is the same for $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{SA}, \mathsf{RA}\}$, so we present it only once, first. The new notions, although somewhat technical, are the natural extensions of relations R_{U} and R_{U}^T to capture indistinguishability of the values instead of equality. Intuitively, in these notions, the adversary does not get to choose the values in the message

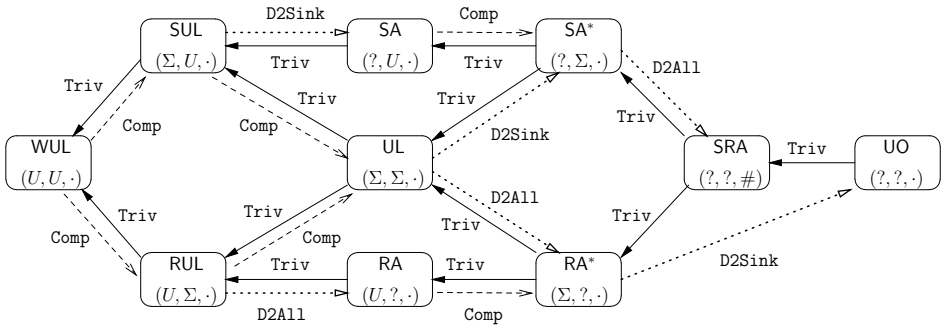

Figure 3.3 Relations among notions of anonymity. Arrows labeled `Triv` denote trivial implications (Proposition 3.4.1) and those labeled `Comp` denote implications under computational assumptions (Lemma 3.4.2). Arrows labeled `D2Sink` and `D2All` denote implications that use the transformation of the same name (Proposition 3.4.6 and Proposition 3.4.7 respectively).

matrices. Instead, the adversary chooses the *distributions* under which those values are drawn before the protocol starts. The adversary, however, does not obtain these values unless leaked by the protocol implementation. Proving that the resulting notion **I**-**N** is in fact *implied* by the original notion **N** is nonetheless non-trivial.

Let $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{SA}, \mathsf{RA}\}$. Given **N**-anonymity, we define notion **I**-**N**-anonymity using an experiment similar to that underlying the definition of **N**-anonymity. In fact, the only difference is that the adversary can specify two PPT *sampling* algorithms $G^{(0)}$ and $G^{(1)}$ from where the elements of the challenge matrices $M^{(0)}, M^{(1)}$ are drawn. The only restriction is that $G^{(0)}$ and $G^{(1)}$ must induce computationally indistinguishable ensembles.[6] Details follow.

Let $k \in \mathbb{N}$ be a security parameter. For simplicity, assume that each party only sends a single message to each other party.[7] Two algorithms $G^{(0)}(\cdot,\cdot)$ and $G^{(1)}(\cdot,\cdot)$ form an *indistinguishable sampling pair* if each is PPT on the first input, and the ensembles $\left\{G^{(0)}(k,a)\right\}_{k\in\mathbb{N},a\in V}$ and $\left\{G^{(1)}(k,a)\right\}_{k\in\mathbb{N},a\in V}$ are computational indistinguishable. We

[6] At first look, this type of adversary may seem artificial, as the restrictions on the sampling algorithms cannot be efficiently tested. Nonetheless, this is all we need, as Proposition 3.4.5 shows any adversary attacking one of the stronger notions can be transformed into this type of adversary, which in turn Proposition 3.4.4 shows can be mapped into an "regular" adversary for the original notion.

[7] The implications still hold if more than one message is exchanged between each pair of parties although the proof becomes a little more involved.

say PPT algorithm A is a *legal* adversary if, on input k, A's first output is a tuple $(M^{(0)}, M^{(1)}, \langle G^{(0)}\rangle, \langle G^{(1)}\rangle)$ where $M^{(0)}, M^{(1)}$ are message matrices and $\langle G^{(0)}\rangle, \langle G^{(1)}\rangle$ is the encoding of an indistinguishable sampling pair. Given a legal adversary A, we define the experiment $\mathbf{Exp}^{\mathbf{I\text{-}N}-anon}_{\pi,A}$ as described below. The corresponding success probability $\mathbf{Adv}^{\mathbf{I\text{-}N}-anon}_{\pi,A}(k)$ of adversary A is defined in the usual way.

Experiment $\mathbf{Exp}^{\mathbf{I\text{-}N}-anon}_{\pi,A}(k)$
  $b \xleftarrow{R} \{0,1\}$, and $(M^{(0)}, M^{(1)}, \langle G^{(0)}\rangle, \langle G^{(1)}\rangle) \leftarrow A(k)$
  if $(M^{(0)}, M^{(1)}) \notin R_{\mathbf{N}}$ **then return** 0
  else Parse $M^{(0)}$ as $(m^{(0)}_{i,j})_{i,j\in[n]}$ and $M^{(1)}$ as $(m^{(1)}_{i,j})_{i,j\in[n]}$
    For all $i, j \in [n]$, all $d = 0, 1$,
      $\bar{m}^{(d)}_{i,j} \xleftarrow{R} G^{(d)}(k, m^{(d)}_{i,j})$ if $m^{(d)}_{i,j} \neq \emptyset$, and $\bar{m}^{(d)}_{i,j} \leftarrow \emptyset$ otherwise.
    $\bar{M}^{(0)} \leftarrow (\bar{m}^{(0)}_{i,j})_{i,j\in[n]}$ and $\bar{M}^{(1)} \leftarrow (\bar{m}^{(1)}_{i,j})_{i,j\in[n]}$
    Execute π on input $\bar{M}^{(b)}$ under adversary A until A outputs a bit g.
    if $(b = g)$ **return** 1 **else return** 0

Definition 3.4.3 Let $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{SA}, \mathsf{RA}\}$. A message-transmission protocol π achieves **I-N**-*anonymity* if for all legal PPT adversaries A, the quantity $\mathbf{Adv}^{\mathbf{I\text{-}N}-anon}_{\pi,A}(k)$ is negligible in $k \in \mathbb{N}$. ▌

We obtain the result of the lemma from the following two propositions. The first one shows that $\mathbf{N} \rightarrow \mathbf{I\text{-}N}$ for any notion $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{SA}, \mathsf{RA}\}$, and the second one proves the results of the lemma starting from **I-N**. Intuitively, this proposition states that the adversary's ability to *choose* the input values for the messages does not significantly weaken the notion of anonymity.

Proposition 3.4.4 Let $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{SA}, \mathsf{RA}\}$, and let π be a message-transmission protocol that achieves **N**-anonymity. Then, π achieves **I-N**-anonymity. ▌

Proof of Proposition 3.4.4: Fix $\mathbf{N} \in \{\mathsf{WUL}, \mathsf{SUL}, \mathsf{RUL}, \mathsf{SA}, \mathsf{RA}\}$. In this case, it is easy to see that for any two message matrices $M^{(0)} = (m^{(0)}_{i,j})_{i,j\in[n]}$ and $M^{(1)} = (m^{(1)}_{i,j})_{i,j\in[n]}$ that belong to relation $R_{\mathbf{N}}$, there exist a permutation $\rho\colon [n]^2 \rightarrow [n]^2$ mapping each pair of indexes (i, j) into another pair $(i', j') = \rho(i, j)$ such that $m^{(0)}_{i,j} = m^{(1)}_{\rho(i,j)} =$

$m^{(1)}_{i',j'}$. (Since such permutation may not be unique, we let $\texttt{Perm}(M^{(0)}, M^{(1)})$ denote the smallest one under some standard encoding.)

Let A be an adversary with non-negligible advantage $\mathbf{Adv}^{\mathbf{I\text{-}N}-anon}_{\pi,A}(k) = \epsilon(k)$. It suffices to show that, either A does not output an indistinguishable sampling pair, or there exist an adversary A^* with non-negligible advantage $\mathbf{Adv}^{\mathbf{N}-anon}_{\pi,A^*}(k)$ that breaks the $\mathbf{N}$-anonymity of π. Indeed, assume we have such A which outputs a sampling pair $\langle G^{(0)}\rangle, \langle G^{(1)}\rangle$. We now show how to build a distinguishing algorithm D for ensembles $\mathcal{X}_0 \stackrel{\text{def}}{=} \{G^{(0)}(k,a)\}_{k,a}$, and $\mathcal{X}_1 \stackrel{\text{def}}{=} \{G^{(1)}(k,a)\}_{k,a}$. Let $D_{i,j}(\cdot)$ be the following algorithm parameterized by $i, j \in [n]$.

Distinguisher $D_{i,j}(x)$

Let $B_{i,j}$ be the following adversary:

Adversary $B_{i,j}(k)$

"Run adversary A, which outputs $M^{(0)}, M^{(1)}, \langle G^{(0)}\rangle, \langle G^{(1)}\rangle$.

Then, define algorithm $H_{i,j}(k,\cdot)$ as follows.

For each $u, v \in [n]$ define $H_{i,j}(k,\cdot)$ as

$$\langle H_{i,j}(k, m^{(1)}_{u,v})\rangle \stackrel{\text{def}}{=} \begin{cases} \langle G^{(1)}(k, m^{(1)}_{u,v})\rangle & \text{for } (u-1)n+v-1 < (i-1)n+j-1 \\ \text{"}\textbf{Output } x\text{"} & \text{for } (u-1)n+v-1 = (i-1)n+j-1 \\ \langle G^{(0)}(k, m^{(1)}_{u,v})\rangle & \text{otherwise} \end{cases}$$

Output $M^{(0)}, M^{(1)}, \langle G^{(0)}\rangle, \langle H_{i,j}\rangle$.

From then on, give any input to A, and output what A outputs.

Once A stops, stop."

return $\mathbf{Exp}^{\mathbf{I\text{-}N}-anon}_{\pi,B_{i,j}}(k)$

We claim that there exists $i^*, j^* \in [n]$, and $a^* \in V$ such that $D_{i,j}$ distinguishes ensembles $\mathcal{X}_0$ and $\mathcal{X}_1$. Wlog. fix the matrices $M^{(0)}, M^{(1)}$ output by A, which we assume belong to relation $R_{\mathbf{N}}$, and thus permutation $\rho = \texttt{Perm}(M^{(0)}, M^{(1)})$ is well defined. Clearly, for all i, j, $\Pr\left[D_{i,j}(G^{(0)}(k, m^{(1)}_{i,j})) = 1\right] = \Pr\left[D_{i',j'}(G^{(1)}(k, m^{(1)}_{i',j'})) = 1\right]$ if $(i'-1)n + j' = (i-1)n + j - 1$. Thus,

$$\epsilon(k) \quad = \quad \mathbf{Adv}^{\mathbf{I\text{-}N}-anon}_{\pi,A}(k)$$

$$
\begin{aligned}
&= 2 \cdot \sum_{i,j \in [n]} \left(\Pr\left[D_{i,j}(G^{(1)}(k, m_{i,j}^{(1)})) = 1 \right] - \Pr\left[D_{i,j}(G^{(0)}(k, m_{i,j}^{(1)})) = 1 \right] \right) \\
&\quad + 2 \cdot \Pr\left[D_{1,1}(G^{(0)}(k, m_{\rho^{-1}(1,1)}^{(0)}) = 1 \right] - 1 \\
&\leq 2 \cdot \sum_{i,j \in [n]} \left| \Pr\left[D_{i,j}(G^{(1)}(k, m_{i,j}^{(1)})) = 1 \right] - \Pr\left[D_{i,j}(G^{(0)}(k, m_{i,j}^{(1)})) = 1 \right] \right| \\
&\quad + \mathbf{Adv}_{\pi, B_{1,1}}^{\mathbf{I\text{-}N}-anon}(k)
\end{aligned}
$$

where we used that $m_{i,j}^{(1)} = m_{\rho(i,j)}^{(0)}$. Notice that $B_{1,1}$ is the adversary that truthfully simulates A, except when A outputs a sampling pair $\langle G^{(0)} \rangle, \langle G^{(1)} \rangle$, in which case $B_{1,1}$ outputs $\langle G^{(0)} \rangle, \langle G^{(0)} \rangle$ instead. We claim that for any such adversary $B_{1,1}$ there exist an adversary A^* (operating in the original experiment) with the same advantage, that is, $\mathbf{Adv}_{\pi, A^*}^{\mathbf{N}-anon}(k) = \mathbf{Adv}_{\pi, B_{1,1}}^{\mathbf{I\text{-}N}-anon}(k)$. Before proving this claim, we show how to obtain the proposition using the claim. In such a case, let $(i^*, j^*) \in [n]^2$ be the indices for which the value in absolute value inside the sum is maximized, and let $a^* = m_{i^*,j^*}^{(1)}$. Then,

$$
\begin{aligned}
\epsilon(k) &\leq 2n^2 \cdot \left| \Pr\left[D_{i^*,j^*}(G^{(1)}(k, a^*)) = 1 \right] - \Pr\left[D_{i^*,j^*}(G^{(0)}(k, a^*)) = 1 \right] \right| \\
&\quad + \mathbf{Adv}_{\pi, A^*}^{\mathbf{N}-anon}(k)
\end{aligned}
$$

Therefore, if $\epsilon(k)$ is non-negligible, then either there exist a distinguishing algorithm $D = D_{i^*,j^*}$ for $\mathcal{X}_0$ and $\mathcal{X}_1$ that succeeds with non-negligible probability on index a^*, or adversary A^* breaks the $\mathbf{N}$-anonymity of protocol π.

We now prove the claim that such A^* exists. Given $B_{1,1}$, we build adversary A^* as follows. Adversary A^* simulates $B_{1,1}$ until the latter outputs $M'^{(0)}, M'^{(1)}, \langle G'^{(0)} \rangle, \langle G'^{(1)} \rangle$. Assume wlog. that $M'^{(0)}, M'^{(1)}$ belong to $R_{\mathbf{N}}$ (otherwise abort) and thus $\rho = \mathtt{Perm}(M'^{(0)}, M'^{(1)})$ is well-defined. Then, A^* computes $\bar{m}_{i,j}^{*(0)} \stackrel{R}{\leftarrow} G^{(0)}(k, i, j)$ and $\bar{m}_{\rho(i,j)}^{*(1)} \leftarrow \bar{m}_{i,j}^{*(0)}$, for all $i, j \in [n]$. The matrices $\bar{M}^{*(0)} = \{\bar{m}_{i,j}^{*(0)}\}_{i,j \in [n]}$ and $\bar{M}^{*(1)} = \{\bar{m}_{i,j}^{*(1)}\}_{i,j \in [n]}$ and then output. From then on, A^* simulates $B_{1,1}$ for the rest of the experiment. It remains to argue that the success probability of $B_{1,1}$, which runs in $E \stackrel{\text{def}}{=} \mathbf{Exp}_{\pi, B_{1,1}}^{\mathbf{I\text{-}N}-anon}$, is as good as that of A^* in $E^* \stackrel{\text{def}}{=} \mathbf{Exp}_{\pi, A^*}^{\mathbf{N}-anon}$. This follows from observing that A^* perfectly simulates $B_{1,1}$ for experiment E^*, so adversary $B_{1,1}$ cannot distinguish whether is executed as part of A^* or inside E. In fact, since $\langle G'^{(0)} \rangle = \langle G'^{(1)} \rangle$, from the point of view of $B_{1,1}$ the distribution of matrix $\bar{M}^{(b)}$ (for any bit b) is identical in both experiments. Since $B_{1,1}$'s

view depends solely on $\bar{M}^{(b)}$, the success probability of $B_{1,1}$ and A^* are thus the same. This concludes the proof of the proposition. ∎

Given any **N**-anonymous protocol π, the simple transformation consisting of encrypting (under an appropriate scheme) each message under the public key of the recipient produces a protocol that achieves the notion **I-N**-anonymity. The next proposition simply shows that breaking the stronger notion gives raise to a *legal adversary* for the weaker notion **I-N**.

Proposition 3.4.5 Assume a semantic secure public-key encryption scheme exists [GM84]. Then I-SUL $\rightarrow$ UL, and I-SA $\rightarrow$ SA*. Moreover, if the encryption scheme is key-private [BBDP01], then I-WUL $\rightarrow$ SUL, I-WUL $\rightarrow$ RUL, I-RUL $\rightarrow$ UL, and I-RA $\rightarrow$ RA*. ∎

Proof of Proposition 3.4.5: We exhibit a simple black-box transformation $\theta^{(\cdot)}$ that, when applied to any **I-N**-anonymous protocol π, where **N** is either WUL,WUL,SUL,RUL, SA, or RA, produces a **N**′-anonymous protocol θ^{π}, where **N**′ is either SUL,RUL,UL,UL,SA*, or RA* respectively. This will prove the desired implications. The construction $\theta^{(\cdot)}$ is simple: given an input set of messages to send, each party encrypts (under the appropriate encryption scheme) each message under the intended recipient's public key, and use those as inputs to π; the local output is then the decryptions of the values received from π. To achieve security, the construction assumes the so-called public key infrastructure (as described in Section 2) in which parties have access to authenticated copies of the public keys for all other parties. Formally, let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a semantic secure encryption scheme [GM84] (which in particular implies $\mathcal{E}$ is randomized) and IK-CPA [BBDP01], and let (pk_i, sk_i) denote the public/private key pair corresponding to party P_i. For any public key pk and message m we denote by $\mathcal{E}(pk, m; r)$ the encryption of m under public key pk using random string r.

We now describe protocol θ^{π} given any message-transmission protocol π. Each party P_i initially holds input $\{m_{i,j}\}_{j\in[n]}$.

1. For each message $m_{i,j}$, each party P_i computes the encryption $y_{i,j} \stackrel{R}{\leftarrow} \mathcal{E}(pk_j, m_{i,j})$

of $m_{i,j}$ under party P_j's public key.

2. Each party P_i, calls protocol π on input $\{y_{i,j}\}_{j\in[n]}$. Let $\{z_{\ell,i}\}_\ell$ be the lexicographically-sorted set that represents the party's local output returned by π.

3. Each party P_i computes the decryption $m'_\ell \stackrel{R}{\leftarrow} \mathcal{D}(sk_i, z_{\ell,i})$ of $z_{\ell,i}$ under using its private key, for all received messages $z_{\ell,i}$.

4. Each party P_i outputs $\{m'_\ell\}_\ell$ as the local output.

The implications stated in the claim are proven next. In what follows, we denote matrices with uppercase letters (say X), and their (i,j)-th elements by lowercase letters (say $x_{i,j}$).

I-WUL $\rightarrow$ SUL: It suffices to show that given protocol $\tau \stackrel{\text{def}}{=} \theta^\pi$ and an arbitrary adversary A_τ attacking the SUL-anonymity of τ, there exists an adversary A_π attacking the I-WUL-anonymity of π. The idea is to let A_π simulate the encryption and decryption phases of protocol τ for A_τ as follows. Adversary A_π on input k, it first executes $A_\tau(k)$. By assumption, $A_\tau(k)$ outputs a pair $(M^{(0)}, M^{(1)}) \in R_{\mathsf{SUL}}$. Adversary A_π then generates a random key pair $(pk, sk) \stackrel{R}{\leftarrow} \mathcal{K}(k)$ and, for $d = 0, 1$, it computes $\langle \hat{G}^{(d)}(k,a)\rangle \stackrel{\text{def}}{=} \langle \mathcal{E}(pk, a; \cdot)\rangle$, where $\langle \mathcal{E}(pk, a; \cdot)\rangle$ denotes the *description* of the probabilistic algorithm that outputs an encryption of a under pk.[8] Adversary A_π then computes new "left-or-right" matrices $\hat{M}^{(0)}, \hat{M}^{(1)}$ by selecting a random value $z \in V$ and then computing $\hat{m}^{(d)}_{i,j} \stackrel{R}{\leftarrow} z$ if $m^{(d)}_{i,j} \neq \emptyset$ and $\hat{m}^{(d)}_{i,j} \stackrel{R}{\leftarrow} \emptyset$ otherwise, for all $i, j \in [n]$ and $d = 0, 1$. The tuple $(\hat{M}^{(0)}, \hat{M}^{(1)}, \langle \hat{G}^{(0)}\rangle, \langle \hat{G}^{(1)}\rangle)$ is then output by A_π. From then on, A_π transparently follows A_π's instructions while attacking π: it forwards all information received from the execution of π to adversary A_τ until A_τ outputs a bit b and stops, in which case A_π outputs the same and stops.[9]

We claim that, unless $\mathcal{AS}$ is not a IND-CPA or IK-CPA secure encryption scheme, A_π correctly simulates the experiment for A_τ. First, notice that the "left-or-right" matrix pair $\hat{M}^{(0)}, \hat{M}^{(1)}$ output by A_π belongs to R_{WUL} if the pair $(M^{(0)}, M^{(1)})$ output

[8] Note that, in the description of the algorithm $\hat{G}^{(d)}$, a denotes a *variable* which is instantiated when the algorithm is evaluated.

[9] Since we do not allow A_τ to corrupt receivers, there is no need to simulate the decryption of the values received by the parties from π. If needed, it would be straightforward though.

by A_τ belongs to R_{SUL}. Now we show that the distribution obtained by the sampling from $\hat{G}^{(0)}, \hat{G}^{(1)}$ during the simulation of A_τ and the distribution of the inputs feed to subprotocol π while running a real execution of τ are computationally close. To see this, let $X = \bar{M}^{(b)}$ be the message matrix used as input to protocol π in $\mathbf{Exp}^{\mathrm{I\text{-}\mathsf{WUL}}\text{-}anon}_{\pi,A_\pi}$; and Y be the message matrix used as input to subprotocol π while executing $\tau = \theta^\pi$ in $\mathbf{Exp}^{\mathsf{SUL}\text{-}anon}_{\tau,A_\tau}$. Clearly, by definition, $x_{i,j} = \mathcal{E}(pk, \hat{m}^{(b)}_{i,j}) = \mathcal{E}(pk, z)$ if $m^{(b)}_{i,j}$ is not empty ($x_{i,j} = \emptyset$ otherwise) for some public key pk and value z chosen anew by A_π, and $y_{i,j} = \mathcal{E}(pk_j, m^{(b)}_{i,j})$ if $m^{(b)}_{i,j}$ is not empty ($y_{i,j} = \emptyset$ otherwise) where pk_j is the public key for party P_j. By a standard hybrid argument, any advantage $\epsilon(k)$ in distinguishing inputs X from Y by A_τ can be transformed into an advantage of at least $\epsilon(k)/(2n^2)$ in breaking the IND-CPA security of the encryption scheme $\mathcal{AS}$, or an advantage of at least $\epsilon(k)/(2n^2)$ in breaking the IK-CPA security of the same scheme. A similar argument shows that A_π outputs a legal sampling pair $\hat{G}^{(0)}(k,a) = \hat{G}^{(1)}(k,a) = \langle \mathcal{E}(pk, a; \cdot) \rangle$ if $(pk, sk) \stackrel{R}{\leftarrow} K(k)$. The proofs for the cases I-$\mathsf{WUL} \to \mathsf{RUL}$, I-$\mathsf{RUL} \to \mathsf{UL}$, and I-$\mathsf{RA} \to \mathsf{RA}^*$ are essentially the same.

For the cases I-$\mathsf{SUL} \to \mathsf{UL}$ and I-$\mathsf{SA} \to \mathsf{SA}^*$ the proof can be done in similar way as above. In these cases, however, it is possible to prove correct simulation of A_τ from *only* the IND-CPA security of the encryption scheme (no IK-CPA security is needed). To illustrate this, we outline the proof of I-$\mathsf{SUL} \to \mathsf{UL}$. (The same proof works for I-$\mathsf{SA} \to \mathsf{SA}^*$.) The simulation is analogous to the one above with the following exceptions: adversary A_π chooses "left-or-right" matrices $\hat{M}^{(0)}, \hat{M}^{(1)}$ by first selecting $z \stackrel{R}{\leftarrow} V$, and then computing, for all $i, j \in [n]$ and $d = 0, 1$, $\hat{m}^{(d)}_{i,j} \leftarrow \langle j, z \rangle$ if $m^{(d)}_{i,j}$ is non-empty and $\hat{m}^{(d)}_{i,j} \stackrel{R}{\leftarrow} \emptyset$ otherwise. Clearly, if $(M^{(0)}, M^{(1)}) \in R_{\mathsf{UL}}$, then $(\hat{M}^{(0)}, \hat{M}^{(1)}) \in R_{\mathsf{SUL}}$. To achieve a correct simulation, A_π sets the sampling pair $\langle \hat{G}^{(0)} \rangle, \langle \hat{G}^{(1)} \rangle)$ to $\langle \hat{G}^{(d)}(k, \langle t, a \rangle) \rangle \stackrel{\text{def}}{=} \langle \mathcal{E}(PK[t], a; \cdot) \rangle$, for $d = 0, 1$, where PK is a table whose index t contains the public key for party P_t. (Notice that each sampling algorithm must include table PK in its description). For the analysis, correct simulation of A_τ by A_π can be easily argued from the IND-CPA security. Indeed, for each column $j \in [n]$, applying sampling algorithm $\hat{G}^{(d)}$ on the (i,j)-th element of $\hat{M}^{(d)}$ generates $\hat{G}^{(d)}(k, \langle j, m^{(d)}_{i,j} \rangle) = \mathcal{E}(PK[j], m^{(d)}_{i,j}; \cdot) = \mathcal{E}(pk_j, m^{(d)}_{i,j})$ which follows the same distribution as the inputs of subprotocol π in an actual execution of τ. Proving that the sampling pair output by A_π is legal is also simpler. Each

"left-or-right" matrix $\hat{M}^{(d)}$, for $d = 0, 1$, contains no duplicate elements per row, therefore each sampling algorithm is guaranteed to be evaluated over different values per row. Therefore, no indistinguishability condition for the sampling algorithms is needed among those indexes – indistinguishability must only hold when evaluated in elements of the same matrix column, say j. In that case, however, the definition of $\hat{G}^{(d)}$, for $d = 0, 1$, guarantees that the same public key pk_j is used, and IND-CPA security suffices to prove the algorithms $(\hat{G}^{(0)}, \hat{G}^{(1)})$ legal. This concludes the proof of the claim. ▌

Proof of Lemma 3.4.2: It follows directly from combining Proposition 3.4.4 and Proposition 3.4.5. ▌

3.4.2 Implications that require "Dummy Traffic"

In this section, we show that notions UL, SA^*, RA^*, SRA, and UO are equivalent under reductions that involve sending dummy traffic. Notions SUL and SA, as well as RUL and RA are also equivalent.

Let `D2Sink` be the following protocol transformation. Given a message-transmission protocol π, output another protocol that operates like π but where each sender transmits additional empty messages *to a fixed party* (the "sink") until the sender's total number equals a given constant. The next proposition shows `D2Sink` can be used to achieve stronger notions of anonymity.

Proposition 3.4.6 Assume the total number of messages in any protocol for the notions $\mathsf{SA}, \mathsf{SA}^*$, and UO is upper bounded by a publicly known value. Then, $\mathsf{SUL}\rightarrow\mathsf{SA}$, $\mathsf{UL}\rightarrow\mathsf{SA}^*$, and $\mathsf{RA}^*\rightarrow\mathsf{UO}$. ▌

Proof of Proposition 3.4.6: The three implications are proven using the same black-box transformation `D2Sink` which maps n-party PPT protocols into other n-party protocols.[10] If applied to any $\mathbf{N}$-anonymous protocol, this transformation (where $\mathbf{N}$ is either SUL,UL, or RA^*) outputs a $\mathbf{N}'$-anonymous protocol (where $\mathbf{N}'$ is either SA,SA^*, or UO respectively). Informally speaking, the construction underlying `D2Sink` relies

[10] `D2Sink` stands for sending "dummy messages to (single) sink".

on "dummy messages". Given as input an arbitrary message-transmission protocol π, $\texttt{D2Sink}$ outputs a protocol $\delta^{\pi}_{\texttt{D2Sink}}$ that essentially operates like π but inputs are "padded" with appropriately-addressed null-valued messages. Indeed, in $\delta^{\pi}_{\texttt{D2Sink}}$, each party's input (which is a set of messages to send) is appended with a certain number of *null-valued messages* whose destination is party P_s, called the "sink", whose identity is fixed for all parties. (Alternatively, P_s can be represented by some non-existent party – the same for all senders – whose traffic gets discarded.) Then protocol π is invoked on the extended inputs which are delivered as expected. Party P_s then discards all null-valued messages it receives. We stress that, in this construction, how to use the "dummy messages" does not depend on the protocol π input to $\texttt{D2Sink}$. The construction does assume, however, that for each notion $\mathbf{N} \in \{\mathsf{SA}, \mathsf{SA}^*, \mathsf{UO}\}$ there exists a quantity $\mu_{\mathbf{N}}$ that bounds the total number of messages that can be sent by any protocol achieving the notion. For concreteness' sake, protocol $\delta^{\pi}_{\texttt{D2Sink}}$ is show next. Here, each party P_i initially holds input vector $(m_{i,j})_{j\in[n]}$.

1. Each party P_i, computes the number of "dummy messages" $\ell_i \leftarrow \mu_{\mathbf{N}} - \sum_{j\in[n]} |m_{i,j}|$ needed.
2. Each party P_i, sets $x_{i,s} \leftarrow m_{i,s} \uplus (\uplus_{i=1\ldots\ell_i}\{\perp\})$, and $x_{i,j} \leftarrow m_{i,j}$ if $j \neq s$.
3. Each party P_i, calls protocol π on input $(x_{i,j})_{j\in[n]}$. Let $\{z_{\ell,i}\}_\ell$ be the lexicographically-sorted multiset that represents local output returned by π to P_i.
4. If $i = s$, party P_i discard any element $z_{\ell,s} = \perp$, and locally output the remaining elements. Otherwise, party P_i outputs $\{z_{\ell,i}\}_\ell$ as the local output.

We now prove $\mathsf{SUL} \rightarrow \mathsf{SA}$. It suffices to show that given protocol $\nu \stackrel{\text{def}}{=} \delta^{\pi}_{\texttt{D2Sink}}$ and an arbitrary adversary A_ν attacking the SA-anonymity of ν, there exists an adversary A_π attacking the SUL-anonymity of π. The idea is to let A_π simulate the operation of protocol ν for A_ν as follows. Adversary A_π on input k, it first executes $A_\nu(k)$. By assumption, $A_\nu(k)$ outputs a pair $(M^{(0)}, M^{(1)}) \in R_{\mathsf{SA}}$. Adversary A_π then generates the appropriate dummy messages for each party P_i by essentially emulating the operation of ν. Namely, for $d = 0, 1$, A_π creates vector $(\hat{m}^{(d)}_{i,j})_j$ from each party P_i's input $(m^{(d)}_{i,j})_{j\in[n]}$

by following steps 1-2 of protocol ν. The pair $(\hat{M}^{(0)}, \hat{M}^{(1)})$ is then output by A_π. From then on, A_π transparently follows A_ν's instructions: it forwards all information received from the execution of π to adversary A_ν and viceversa, until A_ν outputs a bit b and stops, in which case A_π outputs the same and stops. Correct simulation follows from observing that the total number of "dummy messages" sent to P_s is the same no matter what bit b is set in $\mathbf{Exp}^{\mathsf{SUL}-anon}_{\pi,A_\pi}$. By construction, for $d = 0, 1$ the number of messages sent by P_i as instructed by $\hat{M}^{(d)}$ is $f_i = \sum_{j\in[n]} |\hat{m}^{(d)}_{i,j}| = \mu_{\mathbf{N}}$; the total number of messages is then $n\mu_{\mathbf{N}} = \sum_{i\in[n]} f_i = \sum_{i,j\in[n]} |m^{(d)}_{i,j}| + \sum_{i\in[n]} \ell^{(d)}_i$. But since $\sum_{i,j\in[n]} |m^{(0)}_{i,j}| = \sum_{i,j\in[n]} |m^{(1)}_{i,j}|$ then $\sum_{i\in[n]} \ell^{(0)}_i = \sum_{i\in[n]} \ell^{(1)}_i$. Moreover, since all dummy messages sent to P_s are equal to "$\perp$", $(\hat{M}^{(0)}, \hat{M}^{(1)}) \in R_{\mathsf{SUL}}$.

The proof for $\mathsf{UL} \rightarrow \mathsf{SA}^*$ is essentially identical to the one above. The proof for $\mathsf{RA}^* \rightarrow \mathsf{UO}$ is also very similar but slightly more general, as it holds even under adversaries that output message matrices for which $\sum_{i,j\in[n]} |m^{(0)}_{i,j}| \neq \sum_{i,j\in[n]} |m^{(1)}_{i,j}|$, as long as both quantities are upper bounded by a constant μ_{UO}. ▮

Similarly, let `D2All` be the transformation that instructs senders to transmit one dummy message to everyone else per each valid message to be sent. `D2All` is used to prove the following implications.

Proposition 3.4.7 $\mathsf{RUL}\rightarrow\mathsf{RA}$, $\mathsf{UL}\rightarrow\mathsf{RA}^*$, and $\mathsf{SA}^*\rightarrow\mathsf{SRA}$. ▮

Proof of Proposition 3.4.7: The proof follows the same structure as the one of Proposition 3.4.6. Given an arbitrary message-transmission protocol π, protocol $\delta^{\pi}_{\texttt{D2All}}$ works as follows: for each message $m_{i,j}$ in P_i's input, P_i sends a single new null-valued message to all other P_k, $k \neq j$. Then protocol π is invoked on the modified inputs. From the output received by π, each party P_i then discards all received null-valued messages. Let `D2All` be the transform that maps a message-transmission protocol π to another message-transmission protocol $\delta^{\pi}_{\texttt{D2All}}$. Protocol $\delta^{\pi}_{\texttt{D2All}}$ is described next. As opposed to transformation `D2Sink`, this construction does not assume any bounds on the total number of messages exchanged by the parties. Each party P_i initially holds input vector $(m_{i,j})_{j\in[n]}$.

1. Each party P_i, computes the number of "dummy messages" $\ell_{i,j} \leftarrow \sum_{k\in[n]\setminus\{j\}} |m_{i,j}|$ needed to send to party P_j.

2. Each party P_i, sets $x_{i,j} \leftarrow m_{i,j} \uplus (\uplus_{i=1\ldots\ell_{i,j}}\{\perp\})$.

3. Each party P_i, calls protocol π on input $(x_{i,j})_{j\in[n]}$. Let $\{z_{\ell,i}\}_\ell$ be the lexicographically-sorted multiset that represents the local output returned by π to P_i.

4. Each party P_i discard any element $z_{\ell,s} = \perp$, and locally output the remaining elements.

We now prove $\mathsf{RUL} \rightarrow \mathsf{RA}$. Let π be a message-transmission protocol, and $\kappa \stackrel{\text{def}}{=} \delta^{\pi}_{\texttt{D2All}}$. We show that given an arbitrary adversary A_κ attacking the RA-anonymity of κ, there exists an adversary A_π attacking the RUL-anonymity of π. Adversary A_π simulates the operation of protocol κ for A_κ as follows. First, adversary A_π, on input k, obtains a pair $(M^{(0)}, M^{(1)})$ from running $A_\kappa(k)$. From it, A_π generates two new matrices $X^{(0)} = (x^{(0)}_{i,j})_{i,j\in[n]}$ and $X^{(1)} = (x^{(1)}_{i,j})_{i,j\in[n]}$, by adding the appropriate dummy messages for each party P_i according to steps 1-2 of protocol $\delta_{\texttt{D2All}}$ (as described above). Then A_π outputs $(X^{(0)}, X^{(1)})$ as the message matrix pair for experiment $\mathbf{Exp}^{\mathsf{RUL}-anon}_{\pi,A_\pi}$. From then on, A_π transparently follows A_κ's instructions: it forwards all information received from the execution of π to adversary A_κ and viceversa, until A_ν outputs a bit b and stops, in which case A_π outputs the same and stops.

We argue that A_π is a good adversary for $\mathbf{Exp}^{\mathsf{RUL}-anon}_{\pi,A_\pi}$ if A_κ is good for $\mathbf{Exp}^{\mathsf{RA}-anon}_{\kappa,A_\kappa}$. It suffices to show that $(X^{(0)}, X^{(1)}) \in R_{\mathsf{RUL}}$ if $(M^{(0)}, M^{(1)}) \in R_{\mathsf{RA}}$. At this point, we need to define some quantities. For $d = 0, 1$, we denote by $f^{(d)}_i = \sum_{j\in[n]} |m^{(d)}_{i,j}|$ (resp. $\hat{f}^{(d)}_i = \sum_{j\in[n]} |x^{(d)}_{i,j}|$) the total number of messages sent by P_i as encoded by $M^{(d)}$ (resp. $X^{(d)}$). Similarly, $\ell^{(d)}_{i,j}$ denotes the number of "dummy messages" send by P_i to P_j as encoded by $X^{(d)}$, and $\ell^{(d)}_i = \sum_{j\in[n]} \ell^{(d)}_{i,j}$ the total number of such messages. It is easy to see that $\hat{f}^{(d)}_i = \sum_{j\in[n]}(|m^{(d)}_{i,j}| + \ell^{(d)}_{i,j}) = f^{(d)}_i + \ell^{(d)}_i$, and $\hat{f}^{(d)}_i = n \cdot f^{(d)}_i$. So $\ell^{(d)}_i = (n-1)\cdot f^{(d)}_i$. Moreover, since $(M^{(0)}, M^{(1)}) \in R_{\mathsf{RA}}$ then, in particular $(M^{(0)}, M^{(1)}) \in R_{\mathsf{U}}$, which implies $f^{(0)}_i = f^{(1)}_i$, and $\ell^{(0)}_i = \ell^{(1)}_i$, for all $i \in [n]$. Combining these, the multiset of messages

sent by P_i is then

$$\begin{aligned}
\uplus_{j\in[n]} x_{i,j}^{(0)} &= \uplus_{j\in[n]} \left(m_{i,j}^{(0)} \uplus (\uplus_{i=1\ldots\ell_{i,j}^{(0)}} \{\perp\}) \right) = \uplus_{j\in[n]} m_{i,j}^{(0)} \uplus (\uplus_{k=1\ldots\ell_i^{(0)}} \{\perp\}) \\
&= \uplus_{j\in[n]} m_{i,j}^{(1)} \uplus (\uplus_{k=1\ldots\ell_i^{(1)}} \{\perp\}) = \uplus_{j\in[n]} \left(m_{i,j}^{(1)} \uplus (\uplus_{i=1\ldots\ell_{i,j}^{(1)}} \{\perp\}) \right) \\
&= \uplus_{j\in[n]} x_{i,j}^{(1)}
\end{aligned}$$

and $(X^{(0)}, X^{(1)}) \in R_{\mathsf{U}}$ follows.

To argue that $(X^{(0)}, X^{(1)}) \in R_{\Sigma}^{T}$, it suffices to see that the total number of messages ("regular" and and "dummy" messages) to be received by any party P_j according to $X^{(d)}$, $d = 0, 1$, is $\sum_{i\in[n]} |x_{i,j}^{(d)}| = \sum_{i\in[n]} (|m_{i,j}^{(d)}| + \ell_{i,j}^{(d)}) = \sum_{i,j\in[n]} |m_{i,j}^{(d)}| = |M^{(d)}|$. But then, $|M^{(0)}| = |M^{(1)}|$ is implied by $(M^{(0)}, M^{(1)}) \in R_{\mathsf{U}}$, and the result follows.

The proof for $\mathsf{UL} \rightarrow \mathsf{RA}^*$ is analogous (indeed, simpler since we need to prove the matrix pair output by the UL adversary satisfies R_{Σ} instead of R_{U}). A similar argument also proves $\mathsf{SA}^* \rightarrow \mathsf{SRA}$. We notice that, in this latter case, the proof relies on the condition $|M^{(0)}| = |M^{(1)}|$ guaranteed by any SRA-adversary. ∎

3.4.3 Message Overhead and Optimality of the Transformations

The black-box transformations `D2Sink` of Proposition 3.4.6 and `D2All` of Proposition 3.4.7 output protocols that use "dummy" messages (those whose value is "$\perp$" which are ultimately discarded). These messages increase the communication complexity of the protocol, so it is interesting to ask if there are better solutions, possibly based on cryptographic tools. Interestingly, we show that the single transformations `D2Sink` and `D2All` described in previous section cannot be substantially improved, even in the presence of PKI.

Thus, we explore the question of whether more *message efficient* transformations exist, in terms of generating protocols where fewer messages (dummy or not) are sent overall.[11] For simplicity, we consider transformations where the input protocol is

[11] Recall that we say a message m is *sent* by a message-transmission protocol Π if m is an element of the message matrix given to the protocol Π as input. This message should not be confused with

invoked via a black-box call only once; the general case is discussed at the end of the section.

Let $\mathtt{T}$ be a transformation that maps a protocol ω into another protocol $\delta^{\omega}_{\mathtt{T}}$. We measure message overhead by counting the number of extra messages that any protocol $\nu = \mathtt{T}(\omega) = \delta^{\omega}_{\mathtt{T}}$ adds on the underlying (black-box) protocol π. Concretely, given two transformations $\mathtt{T}_1, \mathtt{T}_2$, we would say $\mathtt{T}_1$ has less message overhead than $\mathtt{T}_2$ if protocols $\nu_1 = \mathtt{T}_1(\omega) = \delta^{\omega}_{\mathtt{T}_1}$ and $\nu_0 = \mathtt{T}_0(\omega) = \delta^{\omega}_{\mathtt{T}_0}$ when executed on the same input matrix M require subprotocol ω to send t_1 (resp. t_2) messages when invoked as part of ν_1 (resp. ν_2), where $t_1 < t_2$ for any protocol ω. More formally, let $M = (m_{i,j})_{i,j\in[n]}$ be a message matrix, and denote by $\delta^{[\cdot]}_{\mathtt{T}}(M) \in \mathcal{M}_{n\times n}(\mathcal{P}(V))$ the message matrix on which the black-box protocol (say ω) is invoked via a black-box call during the execution of $\delta^{\omega}_{\mathtt{T}}$ on input matrix M. We stress that once M is fixed, matrix $\delta^{[\cdot]}_{\mathtt{T}}(M)$ is well-defined, independently of the message-transmission protocol ω, as ω is invoked as black-box by $\delta^{\omega}_{\mathtt{T}}$ exactly once.

Definition 3.4.8 Let $(\mathbf{N}', \mathbf{N}) \in \{(\mathsf{SUL}, \mathsf{SA}), (\mathsf{RUL}, \mathsf{RA}), (\mathsf{UL}, \mathsf{SA}^*), (\mathsf{UL}, \mathsf{RA}^*), (\mathsf{RA}^*, \mathsf{SRA}), (\mathsf{SA}^*, \mathsf{SRA})\}$, and $\mathtt{T}$ be any transformation underlying implication $\mathbf{N}' \to \mathbf{N}$. The *message overhead* of $\mathtt{T}$ is $\mathtt{overhead}(\mathtt{T}) \stackrel{\text{def}}{=} \max_M \left\{ \left| \delta^{[\cdot]}_{\mathtt{T}}(M) \right| / |M| \right\}$ where the maximum is taken over all (allowed) non-empty message matrices $|M|$ for notion $\mathbf{N}$. ▌

First, notice that the message overhead of the transformation behind Proposition 3.4.5 is optimal, as no new messages are introduced. Second, it is easy to see that, under the assumption that the total number of messages sent is at most $\mu_{\mathbf{N}}$, $\mathtt{overhead}(\mathtt{D2Sink}) = n \cdot \mu_{\mathbf{N}}$. Similarly, but under no assumptions, $\mathtt{overhead}(\mathtt{D2All}) = n$. The next two propositions show that we cannot do better. The proof is by contradiction which is derived from the fact that if there are "too few" messages sent by a party, the underlying black-box protocol may no longer be invoked in a secure way. For Proposition 3.4.10, the construction and analysis are similar but considering the number of messages *received* by any party.

Proposition 3.4.9 $\mathtt{D2Sink}$ is optimal for $\mathsf{SUL}\to\mathsf{SA}$, $\mathsf{UL}\to\mathsf{SA}^*$, and $\mathsf{RA}^*\to\mathsf{UO}$. ▌

the *packets* sent over the point-to-point communication channels between the parties as the result of a particular implementation of Π.

Proposition 3.4.10 `D2All` is optimal for RUL→RA, UL→RA*, and SA*→SRA. ▌

We now proceed to prove the above propositions.

Proof of Proposition 3.4.9: By contradiction. Assume there exists a transformation $\bar{\mathtt{T}}$ that proves the implication $\mathsf{SUL} \to \mathsf{SA}$ but for which $\mathtt{overhead}(\bar{\mathtt{T}}) < n\mu_{\mathsf{SA}}$. That is, on input any arbitrary SUL-anonymous protocol π, transformation $\bar{\mathtt{T}}$ outputs an SA-anonymous protocol $\bar{\mathtt{T}}(\pi) = \delta^{\pi}_{\bar{\mathtt{T}}}$. Now, let π be a SUL-anonymous protocol and π' be identical to π with the exception that each party P_i also broadcasts the message "sending f_i messages", where f_i is the number of messages that P_i has been instructed to send, that is, $f_i = |\uplus_{j\in[n]} m^{(b)}_{i,j}|$ (where $M^{(b)} = (m_{i,j})_{i,j\in[n]}$ is the corresponding message matrix). Notice that such π' is SUL-anonymous. We then consider the adversary A^*, attacking the SA-anonymity of $\delta^{\pi'}_{\bar{\mathtt{T}}}$, that works as follows. On input $k \in \mathbb{N}$, it outputs two matrices: (a) $M^{(0)}$, which is chosen at uniformly at random among all message matrices with exactly $\sum_{i,j} |m^{(0)}_{i,j}| = \mu_{\mathsf{SA}}$ messages to send. (b) $M^{(1)}$, which contains a single randomly selected row i^* for which $m^{(1)}_{i^*,j} = \uplus_{i\in[n],j} m^{(0)}_{i,j}$, and for all rows $i \neq i^*$, $m^{(1)}_{i,j} = \emptyset$ (that is, in world 1 party P_{i^*} is the only sender but it sends to P_j the same set of messages P_j would receive if it were in world 0). Then, A^* waits for the message "sending f messages" from P_{i^*}: if $f < \mu_{\mathsf{SA}}$ outputs 0, otherwise outputs 1. Then A^* halts.

We argue that A^* is breaks the SA-anonymity of $\delta^{\pi'}_{\bar{\mathtt{T}}}$ with non-negligible probability. Clearly, $(M^{(0)}, M^{(1)}) \in R_{\mathsf{SA}}$. Adversary A^* will distinguish the execution of $\delta^{\pi'}_{\bar{\mathtt{T}}}$ on input $M^{(0)}$ from the one on input $M^{(1)}$ by examining the execution of subprotocol π' on those inputs. To see this, let $X^{(d)} \stackrel{\text{def}}{=} \delta^{[\cdot]}_{\bar{\mathtt{T}}}(M^{(d)})$, for $d = 0, 1$, denote the input matrix for subprotocol π' when $\delta^{\pi'}_{\bar{\mathtt{T}}}$ runs on input $M^{(d)}$. (As usual, we use $x^{(d)}_{i,j}$ to denote the (i,j)-th element of $X^{(d)}$). Assume (for now) that $\left|\delta^{[\cdot]}_{\bar{\mathtt{T}}}(M^{(d)})\right|$ is constant when seen as function of $|M^{(d)}|$. If $\mathtt{overhead}(\bar{\mathtt{T}}) < n\mu_{\mathsf{SA}}$ then it must be the case that $|X^{(d)}| < n\mu_{\mathsf{SA}}$. This, in turn, implies there must exist a sender $P_{i'}$ that sends $\sum_j |x^{(0)}_{i',j}| < \mu_{\mathbf{N}}$ messages using π'. On the other hand, also by assumption, during the execution of $\bar{\mathtt{T}}(\pi') = \delta^{\pi'}_{\bar{\mathtt{T}}}$, all communication is done via π', ie. $\delta_{\bar{\mathtt{T}}}$ is non-interactive. In consequence, P_{i^*}'s input to subprotocol π' is computed by $\delta_{\bar{\mathtt{T}}}$ solely on P_{i^*}'s current input $(m^{(b)}_{i^*,j})_{j\in[n]}$ and random coins, and any publicly known information. It follows that, $|\uplus_{j\in[n]} x^{(1)}_{i^*,j}| \geq |\uplus_{i,j\in[n]} m^{(0)}_{i,j}| = \mu_{\mathsf{SA}}$ (ie. P_{i^*} must send at least μ_{SA} messages via

π') otherwise protocol $\delta_{\bar{\mathsf{T}}}^{\pi'}$ is not a correct message-transmission protocol. Thus, with probability at least $1/n$ (over the choice of i^*), $i^* = i'$, and A^* successfully distinguishes the two executions.

We conclude showing that $\left|\delta_{\bar{\mathsf{T}}}^{[\cdot]}(M)\right|$ is a constant function of $|M|$ if $\bar{\mathsf{T}}$ is a transformation from SUL to SA. The proof is by contradiction. Assume there exists matrices M' and M'' such that $|M'| < |M''| \leq \mu_{\mathsf{SA}}$ but $\left|\delta_{\bar{\mathsf{T}}}^{[\cdot]}(M')\right| < \left|\delta_{\bar{\mathsf{T}}}^{[\cdot]}(M'')\right|$. From the definition of black-box protocol, we know that in protocol $\delta_{\bar{\mathsf{T}}}^{\pi'}$, each sender P_i on input vector $m_{i,*} = (m_{i,j})_{j\in[n]}$ computes a new vector of messages $x_{i,*} = (x_{i,j})_{j\in[n]}$ which is then used as i-th input when calling subprotocol π'. Let us denote this computation by $x_{i,*} = \delta_{\bar{\mathsf{T}}}(m_{i,*})_i$. Thus, in particular, $x'_{i,*} = \delta_{\bar{\mathsf{T}}}(m'_{i,*})_i$, and $x''_{i,*}$ for input $m''_{i,*}$. Since protocol π' is only SUL-anonymous, it must be that $\sum_{j\in[n]} |x'_{i,j}| = \sum_{j\in[n]} |x''_{i,j}|$, for any two inputs $m'_{i,*}$ and $m''_{i,*}$, otherwise protocol $\delta_{\bar{\mathsf{T}}}^{\pi'}$ cannot longer be assumed secure. Moreover, since $\delta_{\bar{\mathsf{T}}}$ is non-interactive, such value $\sum_{j\in[n]} |x'_{i,j}|$ must be constant, say $c_i > 0$. This implies that $\left|\delta_{\bar{\mathsf{T}}}^{[\cdot]}(M')\right| = \sum_{i\in[n]} c_i = \left|\delta_{\bar{\mathsf{T}}}^{[\cdot]}(M'')\right|$ which contradicts the hypothesis. Thus, there exists a constant $c = \sum_{i\in[c]} c_i$, such that $\left|\delta_{\bar{\mathsf{T}}}^{[\cdot]}(M)\right| = c$.

Similar arguments prove the optimality of the transform for the implications $\mathsf{UL} \rightarrow \mathsf{SA}^*$ and $\mathsf{RA}^* \rightarrow \mathsf{UO}$. ∎

Proof of Proposition 3.4.10: We focus on the case $\mathsf{RUL}\rightarrow\mathsf{RA}$ first. The proof is by contradiction. As before, we assume there exists a transformation $\bar{\mathsf{T}}$ that proves the implication $\mathsf{RUL} \rightarrow \mathsf{RA}$ for which $\mathtt{overhead}(\bar{\mathsf{T}}) < n$. That is, on input any arbitrary RUL-anonymous protocol π, transformation $\bar{\mathsf{T}}$ outputs an RA-anonymous protocol $\bar{\mathsf{T}}(\pi) = \delta_{\bar{\mathsf{T}}}^{\pi}$. Now, let π be a RUL-anonymous protocol and π' be identical to π with the exception that each party P_i also broadcasts the message "received g_i messages", where g_i is the number of messages that P_i has received after π has ended, that is, $g_j = |\uplus_{i\in[n]} m_{i,j}^{(b)}|$ (where $M^{(b)} = (m_{i,j})_{i,j\in[n]}$ is the corresponding message matrix). Notice that such π' is RUL-anonymous. Now, if $\mathtt{D2All}$ is not optimal, there exists a transformation $\bar{\mathsf{T}}$ from RUL to RA with $\mathtt{overhead}(\bar{\mathsf{T}}) = \max_M \left\{\left|\delta_{\mathsf{T}}^{[\cdot]}(M)\right| / |M|\right\} < n$. Let M^* the matrix on which the maximum is reached. Then, $\left|\delta_{\mathsf{T}}^{[\cdot]}(M^*)\right| < n \cdot |M^*|$. Notice this implies that there exists a party $P_{i'}$ that receives $\sum_j |x_{i',j}^{(0)}| < |M^*|$ messages using π'.

We then consider an adversary A^* which attacks the RA-anonymity of $\delta_{\bar{\mathsf{T}}}^{\pi'}$. A^* works as follows. On input $k \in \mathbb{N}$, it outputs $(M^{(0)}, M^{(1)})$ that satisfy (a) $M^{(0)} = M^*$, and (b) $M^{(1)}$ contains a single uniformly selected at random column j^* for which $m_{i,j^*}^{(1)} = \uplus_{j\in[n],i} m_{i,j}^{(0)}$, and for all other columns $j \neq j^*$, $m_{i,j}^{(1)} = \emptyset$ (that is, party P_{j^*} receives from P_i all messages sent by P_i in world 0, even those addressed to other recipients). Then, A^* waits for the message "received g messages" from P_{j^*}: if $g < |M^*|$ outputs 0, otherwise outputs 1. Then A^* halts.

We argue that A^* is breaks the RA-anonymity of $\delta_{\bar{\mathsf{T}}}^{\pi'}$ with non-negligible probability. Clearly, by construction, $(M^{(0)}, M^{(1)}) \in R_{\mathsf{RA}}$. We now argue that A^* can distinguish the execution of $\delta_{\bar{\mathsf{T}}}^{\pi'}$ on input $M^{(0)}$ from the one on input $M^{(1)}$ by examining the execution of subprotocol π' on those inputs. For $d = 0, 1$, let $X^{(d)} \stackrel{\text{def}}{=} \delta_{\bar{\mathsf{T}}}^{[\cdot]}(M^{(d)})$ denote the input matrix for subprotocol π' when $\delta_{\bar{\mathsf{T}}}^{\pi'}$ runs on input $M^{(d)}$. (As usual, we use $x_{i,j}^{(d)}$ to denote the (i,j)-th element of $X^{(d)}$). Recall that, $\left|\delta_{\mathsf{T}}^{[\cdot]}(M^*)\right| < n \cdot |M^*|$. In our attack, this implies that, in world 0 there exists a party $P_{i'}$ that receives $\sum_j |x_{i',j}^{(0)}| < |M^*|$ messages using π'. On the other hand, by the non-inactivity of $\delta_{\bar{\mathsf{T}}}$ *after* the call to π', the correctness of $\delta_{\bar{\mathsf{T}}}^{\pi'}$, and since π' is called only once, it follows that, $X^{(1)}$ must satisfy $|\uplus_{i\in[n]} x_{i,j^*}^{(1)}| \geq |\uplus_{i,j\in[n]} m_{i,j}^{(0)}| = |M^*|$, ie. P_{j^*} must receive at least $|M^*|$ distinct messages in π'. Thus, with probability at least $1/n$ (over the choice of j^*), $j^* = j'$, and A^* successfully distinguishes the two executions.

Similar arguments prove the optimality of the transform for the implications $\mathsf{UL} \rightarrow \mathsf{RA}^*$ and $\mathsf{SA}^* \rightarrow \mathsf{SRA}$. ∎

Upper bound per Sender: A similar analysis holds if a bound $\hat{\mu}_{\mathbf{N}}$ on the number of messages *per sender* is assumed instead, for SA and SA*-anonymity. (We stress that the implication $\mathsf{SA} \rightarrow \mathsf{SA}^*$ of Lemma 3.4.2 is preserved under this restriction). In this case the overhead is $n \cdot \hat{\mu}_{\mathbf{N}}$, which is also optimal. This formulation, although more restrictive, can be more suitable for certain applications.[12] From a theoretical point, however, it is not clear if there is any advantage to this formulation over the one presented above.

Single vs. Multiple black-box calls: If consider transformations that output pro-

[12] Upper bounds on the number of messages sent *per party* may help to prevent certain *flooding* attacks against mix nets [GT96, Ser04].

tocols that invoke the input (black-box) protocol more than once, then is it possible to prove that the optimal overhead is n. A protocol δ^π that achieves this is the one that uses a *secure multiparty computation protocol* (eg. [BGW88]) to compute $|M|$ using π as communication channel; then, each party calls ensures it sends $|M|$ messages via π by adding sufficient dummy messages. Even though such a secure multiparty protocol can be computed with constant number of invocations to π [BMR90] (and thus, $\mathcal{O}(n^2)$ messages), it is likely that invoking π more than once will render the resulting protocol impractical.

3.5 On the Anonymity of Previous Protocols

3.5.1 Broadcast Networks

Broadcast channels can be used as a straightforward approach to obtain some form of receiver anonymity [PW87]. In general, the most obvious protocol of transmitting a message over the broadcast channel is trivially RA-anonymous. Blaze et al. [BIK+03] recently suggested a protocol for anonymous routing in the context of wireless networks (*Wireless Anonymous Routing* or WAR). Very roughly, their basic protocol is an adaptation of *onion routing* [GRS96] to broadcast networks. In an onion routing protocol, parties act in three different roles: senders, routers, and receivers. A sender P_s who wants to transmit a message m to receiver P_r, proceeds as follows. First, P_s selects a sequence of parties (called "routers"). This sequence of router identities $\mathcal{R} \subseteq [n]$ is called a "path" for the message. Then P_s computes an *onion* for the message m. An onion for m is a nested encryption of m under encryption keys corresponding to all parties in the path, starting with the receiver's public key, and strategically appending the identities of the routers in the path. For instance, if the path is $\mathcal{R} = \{\ell_1, \ldots, \ell_{t-1}\}$, then the onion of message m for receiver P_r equals the ciphertext $C = \mathcal{E}_{pk_{\ell_1}}(\ell_2 || \mathcal{E}_{pk_{\ell_2}}(\ell_3 || \cdots \mathcal{E}_{pk_{\ell_{t-1}}}(r || \mathcal{E}_{pk_r}(m)) \cdot))$ where $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an encryption scheme and $||$ denotes concatenation. The onion C is then sent to the first router P_{ℓ_1} which decrypts C obtaining a pair (ℓ_2, C'). (Notice that $C' = \mathcal{E}_{pk_{\ell_2}}(\ell_3 || \cdots \mathcal{E}_{pk_{\ell_{t-1}}}(r || \mathcal{E}_{pk_r}(m)) \cdot))$ is still a nested encryption.) In this case, we usually say P_{ℓ_1} "peels the onion" C to obtain C' and new intermediate destination ℓ_2. P_{ℓ_1} then sends C' to P_{ℓ_2}. The process continues until party P_r receives the inner-most

part of the onion, ie. $\mathcal{E}(pk_r, m)$, in which case P_r decrypts it to recover the message m.

The scheme proposed by Blaze et al. specializes this protocol to broadcast channels (in fact, wireless networks) by noticing that, more efficient protocols can be achieved if onions are not transmitted by point-to-point channels but broadcasted to all parties. Indeed, in [BIK+03] Blaze et al. present a modification of the standard onion routing (called the WAR protocol) in which each transmission of an onion C is done via the broadcast channel, and *all receivers* attempt to "peel the onion" (decrypt the ciphertext C). Implicitly in the construction, there is the assumption that only the intended recipient (say P_i) will succeed in the decryption of C and thus only P_i will obtain a valid plaintext C'. Moreover, it is implicitly assumed that P_i is able to tell if C' equals to the message m or if C' is still a nested encryption. In the former case, P_i locally outputs m; otherwise, P_i proceeds to broadcasts C'. See [BIK+03] for details. In this section, we formalize the (implicit) assumptions in [BIK+03] in order to precise the anonymity guarantees provided by their scheme.

Figure 3.4 shows (a slightly simplified version of) the basic protocol by Blaze et al. [BIK+03], denoted π_{WAR}. The simplification lies in that π_{WAR} does not include the "dummy traffic" mechanism outlined in [BIK+03]; this simplifies the analysis (but see discussion at the end of the section) Also, for simplicity, we assume a public key infrastructure (PKI) is already in place.

Protocol π_{WAR} requires an encryption scheme $\mathcal{AE}$. In order to prove the anonymity of this protocol, we first precise some properties the encryption scheme $\mathcal{AE}$ must satisfy. The properties, described below, can be shown to be necessary (for the WAR protocol) to be secure as concrete attacks to the anonymity of π_{WAR} exist if the properties do not hold.[13]

CORRECTNESS: There is a subtle issue on the correctness of the protocol as presented in [BIK+03]. The construction assumes the encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is such that an encryption $C = \mathcal{E}_{pk_i}(m)$ of message m under P_i's public key pk_i can only be decrypted "correctly" by P_i (using P_i's secret key sk_i), and for *any* other party $P_j \neq P_i$ decryption of C under secret key sk_j "will fail" in a detectable way (e.g. outputting $\perp$).

[13] These properties were apparently overlooked in [BIK+03].

Protocol π_{WAR}

Public Inputs: $pk_1, \ldots, pk_n$, where pk_i is the public key of party P_i, $k \in \mathbb{N}$, the security parameter, and $t \in \mathbb{N}$ a fixed integer.
Private Inputs: P_i has input $(m_i, d_i) \in V \times [n]$, and secret key sk_i corresponding to public key pk_i.
Private Output: P_i obtains a message $m \in V$ or the empty message.

(1) In the first round, each party P_i computes an "onion" as follows.

1. First, P_i chooses a (multi)set $S = \{\ell_1, \ell_2, \ldots, \ell_{t-1}\} \xleftarrow{R} [n]^{t-1}$ of indexes for the onion routers.
2. Then, P_i computes $C \leftarrow \mathcal{E}_{pk_{\ell_1}}(\mathcal{E}_{pk_{\ell_2}}(\cdots \mathcal{E}_{pk_{\ell_{t-1}}}(\mathcal{E}_{pk_{d_i}}(m_i))\cdots))$, and broadcast C.

(2) In each subsequent round, each party P_j does: First, receive all broadcast messages. Let $\mathcal{C}$ be such set. Then, for each message $C \in \mathcal{C}$, P_i performs the following steps in parallel:

1. Decrypt C with her own public key, that is, $C' \leftarrow \mathcal{D}_{sk_j}(C)$.
2. If $C' \neq \perp$, that is, $C' \in V \setminus \{\perp\}$, locally output C' and halts.
3. Otherwise, broadcast C''.

(3) If each other party $P_j \neq P_i$ ceases to transmit, P_i outputs the empty message and halts.

Figure 3.4 A simplified version of the WAR protocol [BIK+03].

Next, we formally define this property, which we call *multi-key integrity of the ciphertext* (inspired by the integrity of ciphertext under a single key property of [BN00]).

Definition 3.5.1 Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, $k \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ an integer polynomially related to k, and $p = \{pk_1, \ldots, pk_n\}$ be a set of honestly and independently generated public keys $(pk_i, sk_i) \stackrel{R}{\leftarrow} \mathcal{K}(1^k)$. Scheme $\mathcal{AE}$ achieves *multi-key integrity of ciphertext* (mint-ctxt) if, for any polynomial time adversary A on input p and k that outputs a message m and indexes i, j, the following happens with at most negligible probability: if r is chosen uniformly at random, $\mathcal{D}(sk_j, \mathcal{E}(pk_i, m; r)) \neq \perp$. ∎

Notice that the absence of this property not only affects the correctness of a protocol but also its security. Indeed, suppose there exists an adversary A that is able to generate a message m^* for which an honest sender generates an onion C that, at some point of the routing, it can be decrypted successfully for two different routers (but only one of them is the *intended* router for the onion at that point). Such behavior by the routers is externally observable, and therefore, can allow the adversary to distinguish between an execution where m^* is send and one where it is not.

Key Privacy: We assume the encryption scheme satisfies key-privacy as defined in [BBDP01]. A formal definition appears in Section 2. We remark that key-privacy implies another property in the context of nested encryptions. Assume that nested encryptions of up to t levels are allowed, for $t < n$. The property requires that no router can distinguish between a "peeled" onion (the ciphertext C' resulting from honestly decrypting a valid onion between one and $t - 1$ times), and an "unpeeled" onion (a nested ciphertext as prepared by the original sender), even under an adversarially chosen message m for the onion. It is easy to see that this property follows from key privacy. The property is highly desirable as it prevents the adversary from correlating messages arriving to a corrupted router, foiling some known attacks [BPS00].

Instantiating the Encryption Scheme: We now show there exists a encryption scheme that satisfies multi-key integrity of ciphertexts, and key-privacy. We address the properties one by one, indicating how the encryption scheme satisfies the property or how it can be modified to satisfy it.

Regarding multi-key integrity of ciphertexts, it turns out that, with one exception, it is not known whether these properties are achievable for standard public-key schemes. To the best of our knowledge, the only scheme which can be proven to satisfy this property is the one proposed by Camenisch and Lysyanskaya in [CL05].[14] They proposed a scheme specifically designed to implement *onion routing*, in a provable secure way, so it is not surprising the scheme satisfies the property. The scheme $\mathcal{OR} = (\mathsf{FormOnion}, \mathsf{ProcOnion})$ is specified by two functions FormOnion, ProcOnion. (In what follows, we keep the discussion at high level to avoid cluttering details.) Function FormOnion *creates the onion* by taking the list of public keys of the parties in S, the intended receiver d_k, and a message m_k, and outputting the onion (nested encryption) C. Function ProcOnion is executed by each intermediate router or receiver P_i; this function *processes the onion* by taking as input the onion (nested encryption) C, and the secret key sk_i of the router/receiver and outputs either the plaintext if the party performing the operation was the receiver, or a new "peeled" onion C' if the party performing the operation was a router. How do parties know if they are the intended receivers of the onion? Via an identifier which is appended to the "plaintext" of each onion layer. This identifier specifies the next router in the path. More precisely, in [CL05], each party P_i is publicly assigned an identifier L_i from a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of labels chosen uniformly at random at the beginning of the protocol. Once a sender P has selected a set $S = \{\ell_1, \ell_2, \ldots, \ell_{t-1}\}$ of routers for message m, P prepares the onion by calling the function FormOnion on input the message m, the set of public keys $pk_1, \ldots, pk_n$, and the set of labels $L_{\ell_1}, L_{\ell_2}, \ldots, L_{\ell_{k-1}}$; function FormOnion outputs an onion C. Once the onion C is transmitted to a receiver P_i, the receiver decrypts it via the ProcOnion function. Function ProcOnion returns not only the "plaintext" (possibly still a nested encryption) but also a label L that indicates the *next router* for the onion. For example, if when P_i runs $\mathsf{ProcOnion}(sk_i, C)$ it outputs a plaintext C' and a label $L = L_j \in \mathcal{L}$, then P_i sends the "peeled" onion C' to party P_j; if $L \notin \mathcal{L}$ then the decrypted message C' is assumed to be destined for P_i (see details in [CL05]).

We do not further explain here how functions FormOnion and ProcOnion are

[14] An ElGamal-based scheme by Sako [Sak00] achieves a similar but substantially weaker property. In this scheme, $\mathcal{D}(sk_j, \mathcal{E}(pk_i, m^*; r)) \neq m^*$ holds with negligible probability but only if message m^* is public and fixed in advance.

implemented as it is not needed to understand our security proof (the interested reader is referred to [CL05]). Instead, we point out that the scheme of [CL05] can be seen as a *encryption* scheme $\mathcal{AE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*)$ by identifying encryption function $\mathcal{E}^*$ (resp. decryption function $\mathcal{D}^*$) with FormOnion (resp. ProcOnion) function. Jumping ahead, it will be a variant of encryption scheme $\mathcal{AE}^*$ what we use to securely implement of the WAR protocol.) Camenisch and Lysyanskaya prove a very strong property of the functions FormOnion and ProcOnion called "onion security". Such property suffices for our purposes as it implies the scheme $\mathcal{AE}^*$ satisfies multi-key integrity of ciphertexts.

We now show a simple variant of $\mathcal{AE}^*$ achieves key privacy. Depending on how it is implemented, the encryption scheme $\mathcal{AE}^*$ may not satisfy key-privacy [BBDP01]. There are two reasons for this: first, the encryption scheme used as building block in [CL05] may not guarantee that ciphertexts do not reveal information on the intended recipient of a broadcasted onion. And secondly, the particular implementation of function FormOnion in [CL05] – as outlined before – does reveal the identity of the router to which the onion must be relayed next, and therefore, the last router that process the onion before it reaches the receiver can trivially know the identity of the receiver.[15]

To address the first issue, we note the following. In [CL05], Camenisch and Lysyanskaya use a *IND-CCA secure encryption scheme that support tags* (see [SG02]) to implement functions FormOnion and ProcOnion. If this IND-CCA secure encryption scheme does not satisfy key-privacy, neither does $\mathcal{AE}^*$. (The security proof in [CL05] does assume this IND-CCA secure encryption scheme satisfies key-privacy, even though the requirement is not explicitly made.) Using a standard construction (cf. [SG02]), it is possible to prove that if given a key-private encryption scheme (with no tags) that achieves IND-CCA security then it is possible to build a key-private IND-CCA secure encryption that support tags *as long as the tags do not reveal the receiver*. Key-privacy for the encryption scheme $\mathcal{AE}^*$ follows then by noticing that the construction in [CL05] generates tags that look pseudorandom to anyone other than the final receiver of the onion. Let $\mathcal{AE}^{**}$ denote the encryption scheme implemented as indicated above.

[15] Strictly speaking, this second point is irrelevant under adversaries that do not corrupt parties. We mention the fix only to highlight one of very few details needed to lift the analysis to the case of (passive) corruptions.

We now address the second problem. Consider the following simple variant of scheme $\mathcal{AE}^{**}$: it operates exactly as before (in terms of functions FormOnion and ProcOnion) but *with no initial association between any party and the randomly chosen strings* in $\mathcal{L}$. This means that, even though the sender P picks the labels that identify the routers for the onion C by choosing them uniformly at randomly from the set $\{L_1, \ldots, L_n\}$, the receiver P_i does not use these labels to route the onion. If P_i obtains a string $L_j \in \mathcal{L}$ as part of the onion decryption C', P_i only knows the message is intended for "someone else", and consequently, P_i then broadcasts C'. We denote this variant as $\overline{\mathcal{AE}}$.

We now prove the RA^*-anonymity of the protocol π_{WAR}. We assume there exists an encryption scheme $\mathcal{AE}$ which satisfies semantic security, key-privacy, and multi-key integrity of ciphertexts. The above discussion shows that there exists an encryption scheme, namely $\overline{\mathcal{AE}}$, which satisfies these properties.

Proposition 3.5.2 Assume there exists an asymmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ which satisfies semantic security [GM84], key-privacy [BBDP01], and multi-key integrity of ciphertexts. Then, under passive global adversaries, protocol π_{WAR} achieves RA^*-anonymity. ∎

Proof: The proof is done using the "game-playing" or "sequence of games" argument [BR04, Sho04], although for simplicity we keep the discussion at a high level. Without lost of generality, we set $t = 2$ (the number of "hops" or routers that any onion must travel before the destination is reached). This suffices passive adversaries with no corruptions. Let $pk(P), sk(P)$ denote the public/private key of party P, and let A be a successful adversary breaking the RA^*-anonymity of π_{WAR}. Assume, for simplicity that A only instructs each sender P_i to send a single message $m_{i,j}$ to some destination P_j (ie. A outputs message matrices $M^{(0)}, M^{(1)}$ where $|m_{i,j}^{(b)}| = 1$ for all $i, j \in [n]$, and $b = 0, 1$.)

Notice that if $t = 2$, the execution of protocol π_{WAR} can be seen as composed by three phases. In the first phase, each sender P_i broadcast the nested encryption C_i^1 of her message m_i (ie. C_i^1 is P_i's onion encoding of message m_i). In the second phase, each receiver that correctly decrypts the ciphertext (called a *router* and denoted by $P(C_i^1)$) proceeds to resend the decrypted message $C_i^2 = \mathcal{D}_{sk(P(C_i^1))}(C_i^1)$. In this, we say the router

has "peeled" and then resent the onion. Then, in the third phase, each intended final recipient (denoted $R(C_i^2)$) receives the ciphertexts broadcasted in the previous phase (the once-peeled onions) and obtain the message m_i, which they locally output.

First, fix a bit $b \in \{0,1\}$ and consider a game G_0^b that consists of running A until it outputs a pair of message matrices $M^{(0)}, M^{(1)}$, and then running protocol Π_{WAR} on input $M^{(b)}$ while giving A a copy of all messages exchanged between the parties. Now, consider the game G_1^b that operates as G_0^b but where each router $Q = P(C_k^1)$ resends a *random* message r_Q, instead of the value C_k^2 obtained by decrypting the message broadcasted in the first round. (The fact that game G_1^b does not preserve the correctness of the protocol is irrelevant for passive adversaries with no corruptions.) We argue that both games are indistinguishable for an adversary, as any successful distinguisher for games G_0^b and G_1^b can be used to break either the semantic security or the multi-key integrity of ciphertexts of the encryption scheme $\mathcal{AE}$. As mentioned before, this shows the latter property (multi-key integrity of ciphertexts) is not only needed for correctness but also for privacy, as an adversary could attempt to instruct a sender to encrypt a value m such that the encryption could be decrypted to a meaningful value by two or more receivers, which would help the adversary to succeed in the anonymity experiment. Now, consider the game G_2^b which operates like game G_1^b but where each sender chooses the destination for message $m_{i,j}$ uniformly at random between all parties $P_1, \ldots, P_n$. It is not hard to see that any good adversary distinguishing games G_1 and G_2 can be used to break the key-privacy of the encryption scheme. At this point, notice that the adversary's view for G_2^0 and G_2^1 is identically distributed, since both games are independent of $M^{(0)}$ and $M^{(1)}$. We conclude the proof showing that the distinguishing probability of any adversary A is bounded by the sum of the distinguishing probabilities for the sequence of games $G_0^0 \to G_1^0 \to G_2^0 \to G_2^1 \to G_1^1 \to G_0^1$. Our argument above shows that if the encryption scheme satisfies semantic security, key-privacy, and multi-key integrity of ciphertexts, then this sum of probabilities is negligible. In consequence, the success probability of A must be negligible. ▌

Discussion: As mentioned before, in [BIK+03] a mechanism for "dummy traffic" is used which we omit. The mechanism presented there is randomized, and requires a

party P_i to send a random ("cover") message in step (2) with probability $1/N$ at each round (as long as P_i is not already resending a received onion or sending a message of her own). It is not clear the relevance of such mechanism in our model, as it does not seem to be enough to prove SRA or UO-anonymity. Our results show that the deterministic approach (transform `D2Sink`, which, in this context, can be built as a one-layer onion of a fix message, say $\perp$) suffices to provide stronger guarantees, namely UO-anonymity. However, due to the shared nature of the wireless medium, transforming the WAR protocol into a UO-secure protocol may not be practical given the message overhead (which by Proposition 3.4.9 is unavoidable).

3.5.2 DC-nets or Anonymous Broadcast

DC-nets [Cha88, GJ04] can be seen as particular instances of anonymous broadcast protocols [SA00]. In these protocols, there is a single message sent which is public. In [GJ04], Golle and Juels proposed very efficient anonymous broadcast protocol based on pairings. Whenever a transmission is to take place, all parties participate in the protocol by transmitting "pads". Each pad contains the (potentially empty) message the party intends to transmit. Golle and Juels show how to combine the pads so the transmitted messages are recovered with high probability (and therefore theirs is a message-transmission protocol with high probability). They also show how each party can provide a non-interactive zero-knowledge (NIZK) proof [FLS99] for the correctness of her pad without revealing the underlying message. We claim that their protocol (the one called "short DC-Net") can be proven SA-anonymous under global passive adversaries. Intuitively, since our model does not consider party corruptions, the simulatability of the NIZK proof and the *Decisional Bilinear Diffie Hellman* assumption [BF01] suffices to prove the adversary cannot guess which senders were involved from the pads generated by the parties in the protocol.

Claim 3.5.3 Let π_{DC} be the short DC-Net protocol of [GJ04]. Assume there exist a non-interactive zero-knowledge proof (NIZK) the correctness of each sender's pad in π_{DC} (as described in [GJ04, Section 2.4]) and the Decisional Bilinear Diffie Hellman assumption [BF01] holds. Then, protocol π_{DC} achieves SA-anonymity. ∎

Proof: (Sketch) The proof is a simple extension of the proof of Proposition 1 in [GJ04]. Loosely speaking, the security model used in [GJ04] – which is referred as the *privacy* property there – requires the adversary to identify two parties P_1, P_2 together with a message m and then, after a randomly chosen one of them transmits m using the DC-net protocol, the adversary must guess the sender. The proof in [GJ04] goes essentially like this: given an adversary B attacking the privacy property, Golle and Juels create an adversary D that distinguishes between a Bilinear Diffie-Hellman pair and a *random* Bilinear pair by embedding some elements of the pair the public keys of two random chosen parties $\overline{P_1}, \overline{P_2}$ and some other elements as part of the computation of $\overline{P_1}$'s and $\overline{P_2}$'s pads (if $(P_1, P_2) \neq (\overline{P_1}, \overline{P_2})$ the adversary aborts). Adversary D outputs 1 if B guesses correctly, and 0 otherwise. This adversary D is proven to work because the simulation is perfect if the bilinear pair given to D is a Diffie-Hellman pair, otherwise B should see random pads and has negligible advantage in breaking the privacy property.

We can extend Golle and Juels' proof to our setting by simply using a hybrid argument as follows. Let A be an adversary attacking the SA-anonymity of $\pi_{\mathtt{DC}}$. Given two transmission matrices $M^{(0)}$ and $M^{(1)}$, we create a collection of hybrid matrices $M_0 = M^{(0)}, M_1, \ldots, M_{\ell-1}, M_\ell = M^{(1)}$, where $\ell = |M^{(0)}|$, such that, for $0 \leq i < \ell$, M_i and M_{i+1} differ in the messages sent by two parties, namely in M_i party $\tilde{P}_1$ sends a message $\tilde{m}$ and $\tilde{P}_2$ sends nothing, while in M_{i+1}, $\tilde{P}_1$ sends nothing and $\tilde{P}_2$ sends m. (This is possible as each sender sends at most one message in both $M^{(0)}$ and $M^{(1)}$.) Notice that, by construction of the hybrid matrices, for each i, distinguishing the protocol execution on input M_i from an execution it on input M_{i+1} is exactly (under our adversarial model) what adversary B is assumed to do in the proof of [GJ04]. Therefore, we can simply create a distinguisher D' for the Bilinear Diffie-Hellman problem using the standard hybrid approach: On input a bilinear pair, distinguisher D' first chooses $i \stackrel{R}{\leftarrow} \{0, \ldots, \ell\}$, creates matrices $X_0 = M_i$ and $X_1 = M_{i+1}$, flips a coin $d \in \{0, 1\}$, simulates a run of protocol $\pi_{\mathtt{DC}}$ with input X_d under the adversary A *while following the strategy of [GJ04] and embedding elements of the given pair in the pads*, and outputs 1 if and only if A guesses correctly d. By the usual hybrid argument, it is easy to show that, if A has advantage ϵ in breaking the SA-anonymity of $\pi_{\mathtt{DC}}$, the new distinguisher D has advantage $\epsilon' = c\epsilon/\ell$, where c is the probability the D aborts (while following the

embedding strategy of [GJ04]) which is larger than the inverse of a polynomial on the security parameter k. Since ϵ' is non-negligible, the claim holds. ▮

Notice that this result is not implied by the original security proof in [GJ04] as the anonymity notion used there is arguably different (see Section 3.1.3). Also, even if our model is extended to include passive *corruptions* (see Section 3.6), any corrupted party would send the same messages in both worlds (i.e. matrices $M^{(0)}$ and $M^{(1)}$) so the same argument would work as none of the corrupted parties would be chosen as $\tilde{P}_1$ or $\tilde{P}_2$ in the hybrid argument.

3.5.3 MIX networks

A very popular technique for anonymity providing systems is the use of a MIX network or MIX protocol. Very informally, a MIX network is a protocol that allows several parties to take a vector of encrypted values and "shuffle" it, ie. permute the order of the values in the vector, in a way that leaves the particular permutation hidden to the adversary. More precisely, given a list of ciphertexts, the goal of a MIX network protocol is to output another list of values (either a list of ciphertexts or just plaintexts) which contains the same set of plaintexts as those contained in the encrypted input vector but in a possibly different order. The privacy condition of a MIX protocol requires that no proper subset of the parties (and consequently, no adversary) knows the permutation that maps plaintexts in the input list to the same plaintexts in the output list. There are two main types of MIX networks, re-encrypting and decrypting ones. In the former, the output vector is a re-encryption of the input vector, while in the latter, the output vector contains the decryption of the input vector; in both cases, the order of individual entries in the output vector is potentially different from that of the input vector.

In practice, most MIX protocols work sequentially, where each party (called mixer) acts on the inputs by taking turns, one per party. Each mixer, in order, performs a "shuffle" of the list of ciphertexts, proves the correctness of the shuffle operation – via an interactive protocol–, and then passes the output list to the next mixer. The last mixer broadcasts the result, the output vector. Robust and efficient MIX-net constructions can be built from efficient schemes to *prove a shuffle* [FS01, Gro03, NSNK04]. A

shuffle by mixer P is the operation whereby P, on input a list of ciphertexts, chooses a random permutation, computes the new list of encryptions as re-encryptions of the input list according to the permutation and outputs the resulting list. Decryption MIX networks can be implemented in settings where mixers share decryption keys (eg. as in threshold encryption [DF89]). In those cases, each shuffle may include some form of partial decryption whose correctness is also proven. The advantage of this approach is that the output of the last mixer consists of only plaintexts.

MIX-NET BASED CONSTRUCTION FOR ANONYMITY: A simple construction for anonymity based on mix network techniques is the following. It assumes a threshold encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ (see [GJKR99] and references therein), and a (non-threshold) encryption scheme $\mathcal{AE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$. It also assumes there is a protocol for which a mixer P_i can "prove" the shuffle was correctly performed to another party P_j. Such a protocol is called an *interactive proof system* [Gol01] and it is denoted by $\mathcal{P} = \langle \mathtt{P}, \mathtt{V} \rangle$ where $\mathtt{P}$ is the code run by the prover, say mixer P_i, and $\mathtt{V}$ the code run by the verifier, mixer P_j. If the verifier, after running $\mathtt{V}$ outputs 1 (or "accepts") we say the proof is correct, and if it outputs 0 (or "rejects") we say the proof is incorrect. The construction also assumes a PKI infrastructure where a key pk_k is associated to each party P_k (see Section 2).

The protocol is composed by three phases: (1) key generation, (2) mixing, and (3) threshold decryption. In the key generation, all mixers run a distributed key generation algorithm (DKG) where a key pair for a threshold homomorphic encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is collectively generated, so the secret key sk is shared among the mixers (cf. [GJKR99]) and the public key pk is broadcasted. In the mixing phase, each sender P computes a double encryption of the message $m_{i,j}$ by first encrypting message $m_{i,j}$ under the public key pk_j of the receiver using encryption scheme $\mathcal{AE}'$, and then encrypting the resulting ciphertext under the public key pk of the mixers. Next, senders transmit the doubly encrypted ciphertexts to the first mixer, and the mixing process starts. Sequentially in some predefined order, each mixer P performs the shuffle of the list of ciphertexts, proves its correctness to an arbitrary party P' using the proof system $\mathcal{P} = \langle \mathtt{P}, \mathtt{V} \rangle$, and then passes the resulting list to the next mixer. If at any point, a mixer's P proof is incorrect, all parties ignore the offending mixer's contribution by setting P's

output vector equal to P's input vector, and proceed with the following mixer. The last mixer broadcast the final list, and all mixers decrypt each element of the vector using the threshold decryption protocol $\mathcal{D}$. (In the process, each decrypted element is broadcasted). All potential receivers attempt to decrypt each value. If a receiver P_j succeeds in decrypting a value, P_j outputs the resulting plaintext as local output. We call this protocol $\pi_{\mathsf{MIX}}[\mathcal{AE}, \mathcal{AE}', \mathcal{P}]$.

HVZK Proof of Correct Shuffle: Groth [Gro03] and Furukawa [Fur04, Appendix A] present two shuffles for the ElGamal encryption scheme. They prove the shuffles are *honest verifier zero-knowledge* (HVZK) arguments [Gol01]. Both results assume the underlying encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is homomorphic. At very high level, an homomorphic encryption scheme guarantees there exists operations $+$ and $\cdot$ such that, for any public key pk, $\mathcal{E}(pk, M) \cdot \mathcal{E}(pk, M') = \mathcal{E}(pk, M + M')$ for all messages M, M'. More precisely, if $\mathcal{E} : \mathcal{V} \to \mathcal{C}$ and $\mathcal{E}(pk, M; r) \in \mathcal{C}$ denotes the encryption of plaintext $M \in \mathcal{V}$ under public key pk using randomness $r \in \{0,1\}^k$, we say a scheme is homomorphic if there exist operations $+$ and $\cdot$ in $\mathcal{V}$ and $\mathcal{C}$ respectively, such that for any correctly generated public key pk, $(pk, sk) \leftarrow \mathcal{K}(1^k)$, there exists a random value $r^* \in \{0,1\}^k$ such that for any two messages $M, M' \in \mathcal{V}$, $\mathcal{E}(pk, M; r) \cdot \mathcal{E}(pk, M'; r') = \mathcal{E}(pk, M + M'; r^*)$.

The proofs by Groth and by Furukawa allow a mixer P to prove in zero-knowledge to a honest verifier that two vectors of ciphertexts $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$, each encrypted under some public key pk, encode the same set of plaintexts. More precisely, given a public key pk, generated by $(pk, sk) \stackrel{R}{\leftarrow} K(1^k)$, P can prove that (X, Y) is in L_{Shuffle}. Language L_{Shuffle} contains all tuples $((C_1, \ldots, C_n), (C'_1, \ldots, C'_n))$ for which there exists a permutation π of $[n]$, and a vector of random values $(r_i)_{i \in [n]}$, $r_i \in \{0,1\}^k$, such that $C_i = C_{\pi(i)} \cdot \mathcal{E}(pk, 0; r_i)$ for all $1 \leq i \leq n$, where $0 \in \mathcal{V}$ denotes the null element for the group $(\mathcal{V}, +)$. In what follows, to fix ideas, the reader can assume $\mathcal{P}$ denotes the proof system of [Gro03].

Using the appropriate assumptions on the primitives, protocol $\pi_{\mathsf{MIX}}[\mathcal{AE}, \mathcal{AE}', \mathcal{P}]$ can be proven strong receiver anonymous.

Proposition 3.5.4 Assume $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a semantically secure threshold homomorphic encryption scheme for which there exists an honest-verifier zero-knowledge ar-

gument system $\mathcal{P}$ for language L_{Shuffle}. Also, assume also $\mathcal{AE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is an encryption scheme satisfying key-privacy and multi-key integrity of ciphertexts (as in Definition 3.5.1), and π_{DKG} is a secure distributed key generation protocol. Then, protocol $\pi_{\mathsf{MIX}}[\mathcal{AE}, \mathcal{AE}', \mathcal{P}]$ achieves RA^*-anonymity under passive adversaries and no corruptions. ∎

Proof: (Sketch) Let A be an adversary attacking the RA^*-anonymity of protocol $\pi_{\mathsf{MIX}}[\mathcal{AE}, \mathcal{AE}', \mathcal{P}]$. First, notice that protocol π_{MIX} does not hide which parties are sending messages, so RA^*-anonymity is the best we can hope to prove. We construct an algorithm S with black-box access to A. Algorithm S (for simulator) runs A and simulates all parties' execution of protocol π_{MIX}. We prove our result by showing that, if A is attacks the RA^*-anonymity of π_{MIX} with non-negligible probability, then we can build adversaries $S_{CPA}, S_{DKG}, S_{HVZK}$, and S_{KP} by simple modifications to S so the new adversaries contradict either the semantic security of $\mathcal{AE}$, the security of the distributed key-generation algorithm, the honest verifier zero-knowledge property of $\mathcal{P}$ or the key-privacy property of $\mathcal{AE}'$. Notice that the multi-key integrity of ciphertexts property of $\mathcal{AE}'$ is needed for the correctness of the construction but not for its privacy. In fact, if it does not hold, incorrect decryption of the values output by the threshold decryption protocol may occur, but those can only affect the (honest) party performing the decryption.

Algorithm S works essentially by running experiment $\mathbf{Exp}^{\mathsf{RA}^*-anon}_{\pi_{MIX},A}$, that is, by flipping an unbiased coin b, running A to obtain a pair $M^{(0)}$ and $M^{(1)}$, and honestly simulating all parties in protocol π_{MIX} on input $M^{(b)}$. This simulation includes the generation of all public keys, the computation of the encryptions sent by the senders, the execution of the DKG protocol, the sequential shuffling by the mixers, the threshold decryption, and the final decryption by the receivers. S has no output. Now, we consider the new adversaries and then argue about their advantage probabilities.

- We create an adversary S_{CPA} that attacks the semantic security of $\mathcal{AE}$ by modifying S as explained next. First, recall semantic security of encryption is equivalent to the standard notion of indistinguishability under chosen plaintext attack or IND-

CPA (see Section 2.1.3 for details on the definition and corresponding experiment). Adversary S_{CPA} operates as S with the following difference: right after obtaining the message matrices from A, S_{CPA} chooses a random element $m_{i,j} \in M^{(b)}$, computes the encryption $e_{i,j} = \mathcal{E}'(pk_j, m'_{i,j})$ of $m_{i,j}$ where pk_j is the public key of the receiver P_j, exactly as an honest sender P_i would do at this point. Then, S_{CPA} chooses a random message value $r \in V$ and sends the pair $(e_{i,j}, r)$ to the *left-or-right encryption oracle*, and obtains a target ciphertext C. From then on, simulator S_{CPA} uses C as the ciphertext that party P_i would have provided to the MIX network. The rest of the simulation by S_{CPA} is identical to that of S, except that once A outputs a guess bit g, S_{CPA} outputs 1 if and only if g equals b, the bit chosen initially in the simulation.

Suppose the assumption on all primitives stated on the proposition hold except possibly the semantic security of the threshold encryption scheme $\mathcal{AE}$. First, following [BDPR98] we notice that S_{CPA} is playing the *real or random* experiment, which is essentially equivalent to the IND-CPA notion. By a hybrid argument, S_{CPA} successfully wins this game (guesses whether C contains $e_{i,j}$ or the random value r) with probability equal to $1/\mu$ times the success probability that A breaks the RA^*-anonymity of π_{MIX}, where $\mu \in \mathbb{N}$ is the maximum number of messages sent overall by the senders in the anonymity experiment.

- Distinguishing adversary S_{DKG} attacks the security of the distributed key generation protocol π_{DKG} (see [GJKR99]). Loosely speaking, breaking the security of the DKG protocol amounts to distinguishing the execution of the DKG protocol from the ideal execution where the key is chosen by a trusted centralized entity and a simulator $\mathcal{S}$ provides a "faked" view to the adversary (and therefore $\mathcal{S}$ does not know the shared key). Formally, a DKG distinguisher outputs 1 in the former case and 0 in the latter.

  We build such a distinguisher as follows. Distinguisher S_{DKG} works as S does but instead of simulating the execution of π_{DKG}, it feeds A with its input: either the view of a real execution of protocol π_{DKG} or a simulated view of an execution of a distributed key generation done by a trusted third party. Afterwards, once A

outputs the guess bit g, S_{DKG} outputs 1 if and only if g equals b, the bit chosen initially in the simulation. Notice that the threshold decryption can be perfectly simulated by S_{DKG} as, within the simulation for A, S_{DKG} controls all parties, and any ciphertext can be "decrypted" to the appropriate plaintext sent by the sender.

For the analysis, suppose the assumption on all primitives stated on the proposition hold except possibly the security of the DKG protocol. (Notice this includes the semantic security of encryption scheme $\mathcal{AE}$). In that case, the view provided by any simulator $\mathcal{S}$ to A reveals no more information to A beyond what A can infer from seeing any randomly selected public key of encryption scheme $\mathcal{AE}$, as $\mathcal{S}$ does not have access to the shared secret. Therefore, adversary A's advantage cannot be more than negligible. On the other hand, A's view of the simulation by S_{DKG} is exactly as in π_{MIX}, and therefore S_{DKG}'s advantage if the transcript comes from a real execution of the DKG protocol π_{DKG} must be equal to the probability A correctly guesses bit b, that is, A's advantage in the RA^*-anonymity experiment.

- We create an adversary S_{MIX} that breaks the zero-knowledge property of the proof system $\mathcal{P}$ used to prove the shuffle. More precisely, S_{MIX} distinguishes between a transcript of the interaction between two executions: (1) of a proving mixer P_i and a verifying party P_j after carrying out the interactive proof $\mathcal{P} = \langle \mathtt{P}, \mathtt{V} \rangle$ under input $\mathbf{x} = (C, C') \in L_{\text{Shuffle}}$ and (2) a transcript generated by any algorithm $\mathcal{S}$ on the same input $\mathbf{x}$. Note that prover P knows a *witness* for the proof, that is, the permutation π that maps plaintexts in C to the same plaintexts in C' and the vector of random values $(r_i)_{i \in [n]}$ used to re-encrypt the input vector, while algorithm $\mathcal{S}$ does not know such witness. Distinguisher S_{MIX}, as any distinguisher for the zero-knowledge property of $\mathcal{P}$, takes as input a transcript T of the interaction between a prover and an a verifier, and outputs 1 to indicate the transcript comes from a real interaction and 0 otherwise.

  Distinguisher S_{MIX} works as S with the exception that, instead of (honestly) simulating the entire mixing phase, it picks at a random step $i \in [n]$ in the mixing phase sequence and does as follows. Assume that, at such point, it is the turn of (simulated) party P_i to prove to verifying party P_k that input

$\mathbf{x} = (C^{(i)}, C^{(i+1)})$ is in $\mathcal{L}_{\text{Shuffle}}$. Then, S_{MIX} feeds A with transcript T instead of carrying out the corresponding interaction as S would do (namely, as the parties P_i and P_j would do in an honest execution of the protocol π_{MIX}). The rest of S_{MIX}'s operation is identical to S and concludes with S_{MIX} outputting adversary A's view.

As before, suppose the assumption on all primitives stated on the proposition hold except possibly the security of the interactive proof $\mathcal{P}$. By a simple hybrid argument, it is possible to show that S_{MIX}'s advantage probability is at most $1/n$ the advantage probability of A in the RA^*-anonymity experiment. Indeed, we only need to argue that if the transcripts of all the interactive proofs are generated by a simulator with no knowledge of a witness, then A's advantage inside S_{MIX}'s simulation must be at most negligible (and so it is S_{MIX}'s advantage). This is clear as there is no secret that A can use. Therefore, adversary S_{MIX} must have an advantage probability negligibly close to $1/n$ the advantage of A in the RA^*-anonymity experiment.

- We create an adversary S_{KP} that breaks the key privacy property of encryption scheme $\mathcal{AE}'$ by modifying S as explained below. First, recall that an adversary for the key privacy property receives two (honestly generated) public keys pk_0 and pk_1, sends a message x to the encryption oracle which returns a target ciphertext $c = \mathcal{E}'(pk_b, x)$, upon which the adversary must decide whether c is the encryption of x under pk_0 or pk_1. Also, recall that in protocol π_{MIX} any message $m_{k,\ell} \in M^{(b)}$ sent by P_k to P_ℓ is first encrypted to $e_{k,\ell} = \mathcal{E}'(pk_\ell, m_{k,\ell})$ and then doubly-encrypted using $\mathcal{E}$. We refer to encryption $e_{k,\ell}$ as the $\mathcal{AE}'$-encryption of x.

  To break the key privacy of $\mathcal{AE}'$, we rely on a hybrid argument: first, for a fixed adversary A, we identify the set $\mathcal{Y}$ of all senders that are instructed to transmit the same message m to *different* parties depending on bit b (ie. the recipient P_j of $m \in M^{(0)}$ in "world 0" is different than the recipient $P_{j'}$ of $m \in M^{(1)}$ in "world 1") and we randomly pick one sender $P_i \xleftarrow{R} \mathcal{Y}$; then, given public keys pk_0, pk_1 for the key-privacy experiment, we set up the simulation so party P_j's (respectively $P_{j'}$'s) public key is pk_0 (resp. pk_1). Then, we use the target ciphertext $c = \mathcal{E}'(pk_b, m)$

returned by the key privacy oracle as the $\mathcal{AE}'$-encryption of message m provided by sender P_i the beginning of the simulation. Provided that messages corresponding to senders $P_k \neq P_i$ in $\mathcal{Y}$ are given the appropriate destination (namely, the recipient instructed by $M^{(0)}$ if $k < i$ and the recipient instructed by $M^{(1)}$ if $k > i$), a hybrid argument can be made that any advantage of A in distinguishing the execution of π_{MIX} on input $M^{(0)}$ from an execution on input $M^{(1)}$ can be translated to a distinguishing advantage to break the key privacy of $\mathcal{AE}'$.

(Note: We skip a formalization of the above adversary because, although long and involved, it corresponds to a rather standard hybrid argument and hence it provides no substantial insight into this part of the proof.)

For the analysis, as before, suppose the assumption on all primitives stated on the proposition hold except possibly the key privacy of encryption scheme $\mathcal{AE}'$. Then, by the hybrid argument sketched above, adversary S_{KP}'s advantage in the key privacy experiment is at least $1/\mu$ the advantage of A in the RA^*-anon experiment, where μ is the maximum number of messages senders are instructed to transmit by A during protocol π_{MIX}.

We conclude by noticing that, none of the adversaries mentioned above is guaranteed to work with non-negligible probability, but all of them combined are. This can be easily seen by considering an adversary S^* that it first randomly choose whether to attack the semantic security of $\mathcal{AE}$ (via adversary S_{CPA}), the security of the DKG protocol π_{DKG} (via adversary S_{DKG}), the HVZK property of proof system $\mathcal{P}$ (via adversary S_{MIX}), or the key privacy of $\mathcal{AE}'$ (via adversary S_{KP}), and then calls the corresponding adversary. Clearly, the combined success probability of such adversary S^* is at least the advantage probability of A in the RA^*-anonymity experiment. The result then holds. ∎

In [Gro05], a HVZK argument for correct shuffle *and decryption* for any homomorphic encryption scheme (eg. ElGamal) were also presented. This proof allows to dispense with the threshold decryption and is therefore more efficient.

In [NSNK04], Nguyen et al. present another scheme to prove a shuffle for the Paillier encryption scheme [Pai99]; the proof is secure under a specific notion called

chosen permutation attack IND-CPA$_S$. This notion is also inspired on the indistinguishability of ciphertexts under chosen plaintext attack [BDPR98]. It guarantees that the adversary, upon seen a transcript of a proof system corresponding to a specific shuffle done by a mixer, cannot distinguish the permutation behind the shuffle even if the permutation is known to be one of two (adversarially chosen) permutations. Moreover, the adversary gets to see transcripts of polynomially-many chosen shuffles under chosen inputs vectors. Nguyen et al. also prove the shuffle of [FS01] secure under the same notion. Although weaker than HVZK, this notion seem to suffice for the usual privacy requirements for mix networks. Indeed, our MIX-based construction above can be modified to use such a proof system. It suffices to instantiate the encryption scheme $\mathcal{AE}'$ of protocol π_{MIX} with the Paillier cryptosystem, and the proof system $\mathcal{P}$ with the proof of [NSNK04] or [FS01]. Using a very similar argument as the one shown above, we can obtain an analogous result, namely that the construction is RA^*-anonymous. We do not pursue this variant here as it is almost identical to the construction proven above.[16]

3.6 Variants and Extensions

k-ANONYMITY: Intuitively, a protocol achieves k-anonymity if any adversary trying to determine the sender (resp. receiver) of a message can only narrow the sender's identity down to no less than k possible senders (resp. receivers). The concept (along with efficient constructions) was proposed by von Ahn et al. [vABH03] as a way to improve the efficiency of DC-nets. We can accommodate the notion of k-anonymity in our framework by further restricting the relation $R_{\mathbf{N}}$. For each of the message matrices output by the adversary we require at least k non-empty rows (resp. columns) to capture the restriction to k senders (resp. receivers).

PASSIVE ADVERSARIES WITH CORRUPTIONS: It is possible to extend our framework to consider party corruptions. The adversary would be allowed to passively (statically or dynamically) corrupt senders and receivers, with the obvious restrictions that the local

[16] One important difference is in the way the security proof handles the generation of "fake" transcripts by the simulator *without* knowledge of the secret key for the encryption scheme or the permutation used. Here, the transcripts can be trivially created by querying the *chosen permutation attack* oracle in the definition of IND-CPA$_S$ (see [NSNK04]).

inputs and outputs corresponding to the corrupted parties must be the same in the two message matrices output by the adversary. Note that this conditions immediately hold if the corrupted party that does not send or receive messages and only acts as forwarder (router). The security proofs for the protocols mentioned in previous section carry to this stronger model. Extending our framework beyond passive attacks (i.e. to cope with active adversaries) is currently an open problem.

REFERENCE: Most material in this chapter comes from the manuscript titled "An Indistinguishability-based Characterization of Anonymous Channels", Alejandro Hevia and Daniele Micciancio, and at the time of this writing, in preparation for submission for publication.

4

Simultaneous Broadcast

4.1 Introduction

Broadcast channels allow one or more senders to efficiently transmit messages to be received by all parties connected to a (physical or virtual) communication network. Broadcast is a fundamental communication primitive, both in the design of network communication protocols, and in the area of secure multiparty computation. The main security property characterizing broadcast communication is consistency: the messages received by all players as a result of a broadcast transmission operation are guaranteed to be the same. The problem of achieving consistency when implementing broadcast on top of a point to point network (commonly known as the Byzantine agreement problem) is central not only in cryptography, but also to the area of fault-tolerant distributed computation, and it has received enormous attention (e.g., [LSP82, PSL80, FM85, CR93, CKPS01]).

In secure multiparty computation, it is often desirable that the broadcast channel satisfies some additional properties, besides consistency. In applications where multiple senders can broadcast messages at the same time (e.g., when running in parallel many copies of a broadcast protocol with different senders[1]), it is often important to

[1] Also called *interactive consistency* in [PSL80, BOEY03] and *parallel broadcast* in [HM05]. To avoid confusions, we refer to this operation, multiple senders broadcasting messages at the same time, as *multisender broadcast*.

enforce the *simultaneous* transmission of the messages, so that no sender can decide its broadcast message based on the values broadcast by the other players. Achieving this property, also called *independence*, is not as straightforward as it may seem. In general, naive parallel execution of broadcast protocols does not suffice, nor the more sophisticated round efficient approaches presented in [BOEY03, LLR02b]. Indeed, a common conservative assumption in settings where (multisender) broadcast channels are provided is to assume *rushing* adversaries – adversaries that, at each round, may see the messages sent by the honest parties before sending out the messages for the corrupted parties for the same round [BGW88, BMR90]. The independence property plays a fundamental role in the secure multiparty computation protocol of [CGMA85] as well as many important applications (like contract bidding, coin flipping, and electronic voting schemes, as exemplified in [CR87, DDN01, Gen00]) where simultaneous broadcast enormously simplifies the design of the protocols as broadcast is used in a more or less direct way.

The concept of simultaneous broadcast (also called independent broadcast) was first put forward by Chor et al. [CGMA85] who proposed a simulation-based definition, and presented protocols that securely implement simultaneous broadcast on top of a network which allows regular broadcast transmission operations, not necessarily satisfying the simultaneity property. The protocols in [CGMA85] require (for each simultaneous broadcast operation) a number of rounds that is linear in the number of parties. Given the importance of the simultaneous broadcast primitive, subsequent research efforts [CR87, Gen00] focused on reducing the round complexity, obtaining simultaneous broadcast protocols that run in logarithmically many [CR87] or even constant [Gen00] number of rounds (the latter result achieved in the common random string model.) Unfortunately, a close inspection of [CGMA85, CR87, Gen00] reveals that the definitions of simultaneous broadcast used in the three papers are quite different. Although, at first sight, all three definitions may appear appealing and intuitive, the technical differences among them bring up the following questions: what is the relation between the different definitions? Are they equivalent? Are they increasingly stronger or weaker? Or are they perhaps incomparable, in the sense that no one implies the other?

Motivated by the efficiency improvement achieved by [CR87, Gen00] over the original linear round protocol of [CGMA85], we investigate and compare the definitions

proposed in these three papers. (More precisely, we compare their straightforward generalizations to arbitrary input distributions[2].) Informally, our findings rank the original definition [CGMA85] as the strongest, and the most recent definition [Gen00] as the weakest. Technically, we prove implications and separations showing that the original definition [CGMA85] is strictly stronger (in a precise sense to be defined) than the definition of [CR87], which, in turn, is strictly stronger than the latest definition of [Gen00]. The comparison is not so straightforward because not all definitions are achievable for any input distribution, and for any pair of definitions (say, definition A and B) it may be possible to find a protocol Π and a distribution D such that Π is satisfies definition A but not definition B on input drawn according to D. So, it may seem that the definitions are incomparable. In order to properly rank the definitions, we first characterize the class of achievable input distributions for each definition. Our characterization is tight: for each definition A, we give a class of distributions $\mathcal{D}(A)$ such that

- definition A can be achieved in a strong sense: there exists a single protocol Π that satisfies A for any distribution in $\mathcal{D}(A)$
- the class $\mathcal{D}(A)$ cannot be extended even in a weak sense: for any distribution outside $\mathcal{D}(A)$, no protocol can possibly satisfy definition A.

It turns out that the class of distributions associated to the three definitions form a monotonically decreasing sequence. Let Sb, CR and G stand for the definitions given in [CGMA85], [CR87] and [Gen00] respectively, and let $\mathcal{D}(\mathsf{Sb})$, $\mathcal{D}(\mathsf{CR})$ and $\mathcal{D}(\mathsf{G})$ be the corresponding classes of input distributions. We show that

$$\mathcal{D}(\mathsf{Sb}) \supset \mathcal{D}(\mathsf{CR}) \supset \mathcal{D}(\mathsf{G}).$$

Armed with this characterization of the input distributions associated to each definition, we prove implications and separations between the three definitions as follows.

We prove that definition Sb implies definition CR in the sense that for any protocol Π, if Π is Sb-Independent for every distribution $D \in \mathcal{D}(\mathsf{CR})$ (i.e., for any

[2] At the time the definitions were suggested, a prime application of simultaneous broadcast was distributed coin flipping. Apparently influenced by that, the definitions of [CR87, Gen00] were implicitly understood to be used with uniform input distributions even though no such restriction was stated on the original papers.

distribution for which definition CR is achievable at all), then Π is also CR-Independent for every such distribution. Moreover, we give a simple example showing the reverse implication does not hold true, i.e., there exists a class of input distributions (such that Sb-Independence is achievable) and a protocol Π such that Π is CR-Independent but not Sb-Independent for every distribution in that class. We conclude that CR-independence is strictly weaker than Sb-Independence.

Next we prove that definition CR implies definition G in the sense that for any protocol Π, if Π is CR-Independent for any distribution $D \in \mathcal{D}(\mathsf{G})$, then Π is also G independent for any such input distribution. Moreover, we prove that the reverse implication is not true, i.e., there is a protocol Π that satisfies G-Independence for any distribution in $\mathcal{D}(\mathsf{G})$, but it does not satisfy CR-Independence for any nontrivial distribution (including the uniform). We conclude that G-independence is strictly weaker than CR-Independence.

We remark that while the relation between Sb-Independence and CR-Independence was to be expected because Sb resorts to a general secure multiparty computation definitional framework, the relation between CR-Independence and G-Independence was not as clear. In particular, [Gen00] seemed to suggest that the use of statistical notion of independence makes definition G stronger than CR, which uses a computational notion of closeness between distributions. Our results show that when restricted to an appropriate class of distributions, the relation between the two definitions is opposite to the one suggested in [Gen00].

We also remark that while simulation-based definitions are usually stronger than other definitions, and in many other cases in cryptography definitions have been made stronger and stronger over time, to culminate with a definition based on the simulation paradigm, the simultaneous broadcast problem studied in this chapter represents an interesting case in which the reverse process has occurred: the original and strong simulation-based definition has been made weaker and weaker over time in order to achieve greater efficiency. We leave it as an open problem to find a constant-round protocol (i.e., as efficient as the one of [Gen00]) for simultaneous broadcast that achieves the stronger notion of CR-Independence [CR87] or even (and preferably) Sb-Independence [CGMA85].

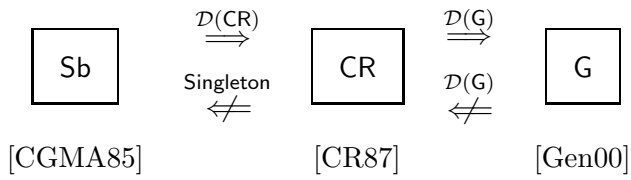

Figure 4.1 Our results. An arrow $\stackrel{\Delta}{\Longrightarrow}$ from definition A to B means that any protocol that achieves definition A under all distributions in Δ also achieves definition B under the same distributions. A broken arrow $\stackrel{\Delta}{\not\Longrightarrow}$ from A to B indicates that the implication $A \stackrel{\Delta}{\Longrightarrow} B$ is false.

Using Arbitrary Input Distributions: The question of whether security can be achieved under input distributions other than the uniform is not only of theoretical interest (comparing definitions) but of very practical relevance. In many applications (like electronic voting or contract bidding), the parties' input are not necessarily uniform or independent from each other – some partial knowledge of the inputs may have leaked. More general input distributions allow us to capture these cases. As a consequence, whether or not a definition of security can be achieved under more general input distributions can determine whether or not a given solution suffices for a particular application (e.g. whether the protocols suggested in [CR87, Gen00] guarantee security in scenarios with partial knowledge of the inputs, like voting). Given that the original definitions in [CR87, Gen00] did not explicitly excluded non-uniform input distributions, we see this contribution as useful in practice. Our characterization of the distributions associated to the definitions of [CR87] and [Gen00] show that those definitions are of limited applicability, as they can be achieved only for a restricted class of input distributions.

4.1.1 Discussion and Related Work

In [DDN01], Dolev et al. introduce the notion of malleability of protocols, and present definitions for non-malleable message encryption, string commitment and zero-knowledge proofs. Loosely speaking, a protocol run by honest party P on private input x is *non-malleable* if no corrupted player P' can use (transform) the execution of the protocol to generate a valid execution of the same protocol under some input x' related to x. Therefore, non-malleability does guarantee some form of independence of the private

values used in different protocols. The results of [DDN01], however, focus mostly on two-party protocols so their definitions do not capture the subtleties underlying the definition of independence of multisender broadcast protocols with more than two players. Along the same line, also in the two party setting, Liskov et al. [LLM+01] study *mutually independent commitments* whose goal is to ensure the "independence" of the committed values. They give definitions which seem to capture – in a strong sense – this property. Their definitions, however, do not immediately extend to the multiparty case.

Organization: The chapter is organized as follows. Section 4.2 presents some terminology, and Section 4.2.1 describes the system model, including the definition of multisender broadcast. In Section 4.3, we present the definitions of independence existing in the literature, and in Section 4.4, a characterization of the sensible input distributions that can be associated to the definitions is made. Then, Section 4.5 concludes with implications and separations between the notions.

4.2 Preliminaries

In this section, we describe some of the basic elements used in this work. We first describe the network model and then we formalize the concept of multisender broadcast.

4.2.1 The Model

In this chapter, we use the network model outlined in Section 2.2.1, namely synchronous network with point-to-point channels between all parties and *rushing* adversary. We remark that our choice of network and adversary model is made mostly to fix ideas, since the model is rather orthogonal to the main focus of the chapter, the definition of independence. Towards this end, we formalize the notion of multisender broadcast in the next section.

4.2.2 Multisender Broadcast

Intuitively, a multisender broadcast protocol is a broadcast protocol that allows all parties to broadcast values at the same time. Notice that, here, the term "multi-sender" refers to the property that multiple broadcast senders are allowed in the same protocol execution. The simplest instantiation of a multisender broadcast protocol is the protocol that performs n sequential executions of a standard (single-sender) broadcast protocol, where in the i-th execution party P_i acts as the sender.

Formally, assume each player P_i has an input bit x_i, and a security parameter k. (Henceforth, for simplicity, we consider the broadcast messages as bits). Consider a protocol Π run by the parties, at the end of which each honest party P_i outputs an n-dimensional vector $\mathbf{B}_i = (B_{i,1}, B_{i,2}, \ldots, B_{i,n}) \in \{0,1\}^n$. Protocol Π is said to implement *multisender broadcast* if it satisfies the following two properties:

(1) **Consistency**: For any adversary A, every honest parties P_i and P_j, $\mathbf{B}_i = \mathbf{B}_j$ with overwhelming probability.

(2) **Correctness**: For any adversary A, every honest parties P_i and P_j, $B_{i,j} = x_j$ with overwhelming probability.

The notion of multisender broadcast was introduced by Pease et al. in [PSL80] where it was called *interactive consistency*.

For every protocol that implements multisender broadcast it is possible to associate a single value to each party as the value *announced* by the party.

Definition 4.2.1 Assume parties $P_1, \ldots, P_n$ run some multisender broadcast protocol Π on input vector $\mathbf{x}$ under some polynomial-time adversary A. Then, for each $i \in \{1, \ldots, n\}$, we define the value "announced" by party P_i as the i-th bit output by any honest party P_k, namely $W_i \stackrel{\text{def}}{=} B_{k,i}$.[3] By the consistency property, the n-dimensional vector $\mathbf{W} = (W_1, \ldots, W_n)$ is well-defined with overwhelming probability. For notational convenience, we let $\text{ANNOUNCED}_A^{\Pi}(\mathbf{x})$ denote vector $\mathbf{W}$ "announced" by the parties after running protocol Π under adversary A on input $\mathbf{x} \in \{0,1\}^n$. Similarly, $\text{ANNOUNCED}_A^{\Pi}(\mathcal{X})$

[3]By convention, if a corrupted party P contributes with an invalid input or no input at all, honest parties assign the default value 0 as the bit "announced" by P.

denotes the induced distribution on $\textsc{Announced}_A^{\Pi}(\mathbf{x})$ when $\mathbf{x}$ is chosen according to some distribution $\mathcal{X}$. ▮

We remark that a multisender broadcast protocol does not necessarily guarantees independence of any sort – the announced values can be correlated even if the inputs are not. For example, the simplest instantiation described before (where n single-sender broadcasts are executed sequentially) satisfies both consistency and correctness but breaks independence: a dishonest last sender P_n could discard its own input and broadcast one of the values previously heard (say, the one broadcast by party P_i). In this case, the i-th and n-th entry in the vector of announced values will always be the equal, no matter the inputs. More sophisticated multisender protocols like the expected constant-round interactive consistency protocol of Ben-Or and El-Yaniv [BOEY03] do not guarantee independence either.

4.3 Simultaneous Broadcast: Notions of Independence

Informally, a protocol Π is said to implement *simultaneous broadcast* (SB) if Π implements multisender broadcast where the values announced are "independent" of each other. Intuitively, the independence property sought must guarantee that no group of corrupted parties may announce values which may somehow depend on the values announced by any subset of the uncorrupted parties. In this section, we review some of the notions of independence previously proposed in the literature.

4.3.1 Chor, Goldwasser, Micali and Awerbuch's definition

In their seminal paper [CGMA85], Chor et al. define simultaneous broadcast as a network property that can be *emulated* starting from a network which provides a broadcast channel. Loosely speaking, Chor et al. show how to build a "compiler" that transforms protocols in a simultaneous broadcast network into protocols in a regular (non-simultaneous) broadcast network such that whatever an adversary can do in the latter network, there exists some adversary that can do the same in the former network.

EXTRACTING A SIMULATION-BASED DEFINITION: We adapt the definition of [CGMA85] to the framework of secure function evaluation of [Can00a] as follows. The case in which the parties have access to a simultaneous broadcast network is cast as the "ideal" process of Canetti's framework [Can00a]. There, all parties have access to a trusted third party which computes the function $f_{SB}(\mathbf{x}) = (\mathbf{x}, \ldots, \mathbf{x})$. In the notation of [Can00a], we call this protocol $\mathsf{Ideal}(f_{SB})$. On the other hand, to capture a regular (non-broadcast) network, we consider a "real" process in which a protocol Π is executed in a partially synchronous network under adversary A. Here, $\text{EXEC}_A^{\Pi}(k, z, \mathbf{x})$ denotes the $(n+1)$-dimensional vector formed by the output of adversary A and the parties after executing protocol Π in the real process with inputs z and $\mathbf{x}$ respectively, and $\text{EXEC}_S^{\mathsf{Ideal}(f_{SB})}(k, z, \mathbf{x})$ denotes the corresponding vector of outputs after $\mathsf{Ideal}(f_{SB})$ is executed with ideal adversary S in the ideal process (see Section 2.2.3 for details). Independence is then captured by requiring that Π securely implements f_{SB} in the sense of [Can00a]. Thus, we obtain the following definition

Definition 4.3.1 [Sb-Independence] Protocol Π achieves Sb-independence if for any PPT adversary A corrupting up to $t < n$ parties, there exists a PPT simulator S such that, the ensembles (indexed by $k \in \mathbf{N}$, $\mathbf{x} \in \{0,1\}^n$, and $z \in \{0,1\}^*$),

$$\begin{aligned} \text{EXEC}_A^{\Pi} &\stackrel{\text{def}}{=} \left\{\text{EXEC}_A^{\Pi}(k, z, \mathbf{x})\right\} \\ \text{EXEC}_S^{\mathsf{Ideal}(f_{SB})} &\stackrel{\text{def}}{=} \left\{\text{EXEC}_S^{\mathsf{Ideal}(f_{SB})}(k, z, \mathbf{x})\right\} \end{aligned}$$

are computationally indistinguishable. ∎

USING INPUT DISTRIBUTIONS: We also consider an alternative simulation-based definition which explicitly involves input distributions. This new definition, described next, is called (All, Sb)-Independence and it is shown to be equivalent to Sb-Independence in Section A.1.1.

Definition 4.3.2 [(Δ, Sb)-Independence] Let Δ be a class of input distribution ensembles over n-bit strings. Protocol Π achieves (Δ, Sb)-independence if for any PPT adversary A corrupting up to $t < n$ parties, there exists a PPT simulator S such that for every distribution ensemble $\mathcal{D} \in \Delta$, the ensembles (indexed by $k \in \mathbf{N}$, and $z \in \{0,1\}^*$)

$$\text{XEXEC}_A^{\Pi} \stackrel{\text{def}}{=} \left\{\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)} \,:\, \left(\mathbf{x},\, \text{EXEC}_A^{\Pi}(k, z, \mathbf{x})\right)\right\} \tag{4.1}$$

$$\text{XEXEC}_S^{\mathsf{Ideal}(f_{SB})} \stackrel{\text{def}}{=} \left\{ \mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)} \ : \ \left(\mathbf{x},\ \text{EXEC}_S^{\mathsf{Ideal}(f_{SB})}(k, z, \mathbf{x}) \right) \right\} \tag{4.2}$$

are computationally indistinguishable. In this case, we say Π is Sb-Independent under class Δ. If $\Delta = \mathsf{All}$, the class of all input distributions over n-bit strings, then we say Π achieves $(\mathsf{All}, \mathsf{Sb})$-Independence. ▮

4.3.2 Chor and Rabin's definition

Chor and Rabin [CR87] proposed another definition of independence. Intuitively, their definition seems to come from the following idea. Let A be an adversary not corrupting party P_i. Any computable information on the $n-1$ bits announced by any party other than P_i can be cast as a (polynomial-time) predicate R on those bits. After fixing the adversary, whether or not this predicate is true defines an event. Then, if the bit output by P_i is probabilistically independent of any such event, then the output of P_i is effectively oblivious (unaffected) by the actions of adversary, thus guaranteeing some independence. A formal definition follows, slightly generalized to consider input distributions. The definition of [CR87], which was presented in a different but equivalent formulation (see Section A.1.2), is obtained as a special case when the input distribution is uniform.

Definition 4.3.3 (CR-Independence) Let $\mathcal{D}$ be an input distribution over $\{0,1\}^n$. A protocol Π achieves CR-independence under input distribution $\mathcal{D}$ if, for any adversary A, all honest party P_i, all polynomial-time predicate R, the quantity

$$\left| \Pr\left[\, \mathbf{W}_i = 0 \,\right] \cdot \Pr\left[\, R(\mathbf{W}_{\overline{\{i\}}}) \,\right] - \Pr\left[\, \mathbf{W}_i = 0 \ \wedge \ R(\mathbf{W}_{\overline{\{i\}}}) \,\right] \right| \tag{4.3}$$

is negligible (in the security parameter k) when $\mathbf{W} \leftarrow \text{ANNOUNCED}_A^{\Pi}(\mathcal{D}^{(k)})$. ▮

4.3.3 Gennaro's definition

The third definition of independence considered here was presented by Gennaro in [Gen00].[4] Loosely speaking, a protocol achieves independence under this definition

[4] A different definition was originally described in a preliminary version [Gen95]. Since such definition evolved into the one of [Gen00], we do not considered it in this work.

if the bit announced by each corrupted party is not correlated with the bits announced by all the honest parties. In [Gen00], it is (implicitly) assumed the inputs to the parties follow the uniform distribution. Below, we slightly generalize the definition of [Gen00] to consider arbitrary input distributions.

Definition 4.3.4 (G-Independence) Let $\mathcal{D}$ be an input distribution over $\{0,1\}^n$. A protocol Π achieves G-independence under input distribution $\mathcal{D}$ if, for all adversaries A corrupting a subset B of parties (where $|B| = t < n$), for each corrupted party P_i, for all bit $b_i \in \{0,1\}$, and for all vectors $\mathbf{r}, \mathbf{s} \in \{0,1\}^{n-t}$ that occur with non-zero probability as $\mathcal{D}_{\overline{B}}$, the quantity

$$\left|\Pr\left[\mathbf{W}_i = b_i \mid \mathbf{W}_{\overline{B}} = \mathbf{r}\right] - \Pr\left[\mathbf{W}_i = b_i \mid \mathbf{W}_{\overline{B}} = \mathbf{s}\right]\right| \tag{4.4}$$

is negligible (in the security parameter k) when $\mathbf{W} \leftarrow \text{ANNOUNCED}_A^{\Pi}(\mathcal{D}^{(k)})$. ▮

A RELATED, SIMPLER DEFINITION: The idea behind the definition of [Gen00] is that, the probability that a corrupted party P_i outputs a bit b_i in the probability space where honest parties end up outputting a vector $\mathbf{r}$ must be about the same for any vector $\mathbf{r}$. This approach may lead to technical difficulties when proving properties of the definition over arbitrary distributions, since the definition may involve conditioning over possibly negligible events. To overcome this problem, we define a related (and possibly stronger) definition which implies Definition 4.3.4. The new definition, called G**-Independence, is presented and shown to imply G-Independence in Section A.1.3. The fact that this new definition implies G-Independence will suffice to show implications and separations with respect to the other notions considered in this work.

4.4 The Role of the Input Distributions

The original definition of [CGMA85], although informal, is based on a general simulation paradigm and is arguably the strongest: a simultaneous broadcast protocol is a protocol that securely computes a function $f(x_1, \ldots, x_n)$ that on input n values $x_1, \ldots, x_n$ (provided by the n protocol participants) returns to each player the vector $\mathbf{x} = (x_1, \ldots, x_n)$ containing all the input values. Part of the power of this definition

comes from the fact that security is required for any fixed input $(x_1, \ldots, x_n)$. This allows to model arbitrary input probability distributions, partial information about the inputs, etc.

In contrast, the definitions proposed in [CR87, Gen00] consider a specific input distribution and are statistical in nature: motivated by coin flipping applications, the definitions of [CR87, Gen00] consider the execution of the protocol when the input values $x_1, \ldots, x_n$ are chosen independently and uniformly at random, and propose a formalization of the intuitive requirement that

- the value broadcast by any honest party is independent from all other broadcast values [CR87], or
- the value broadcast by any corrupted party is independent from the values broadcast by all honest parties [Gen00].

Moreover, the notion of independence used in [CR87] is computational (i.e., it is only required that no polynomial time observer can detect dependencies), while the notion considered in [Gen00] is information theoretic. Both definitions can be generalized to arbitrary input distributions, but the generalization immediately highlights the limitations of the definitions in [CR87, Gen00]: if the input values $x_1, \ldots, x_n$ are strongly correlated, then the desired (correct) output also need to be correlated, and no protocol can possibly achieve the definition. In other words, there are probability distributions for which no protocol can possibly achieve the definitions in [CR87, Gen00]. At the same time, there are trivial distributions (e.g., any singleton distribution that concentrates all probability on a single input vector) for which any protocol vacuously satisfies the definition of [CR87, Gen00]. In other words, there are distributions for which the definitions of [CR87, Gen00] are not meaningful.

In this section, we formalize this intuition and for each definition of independence, we identify the largest class of distributions under which the definition is "achievable". More precisely, for each notion of independence, we prove there is a class of distributions under which the definition of independence can be realized – there exist a protocol that achieves the notion under such a class – but whose complement is not

achievable in a strong sense: no protocol achieves the notion even for a single distribution outside the class. For any definition $\mathsf{N} \in \{\mathsf{CR}, \mathsf{G}\}$, we say a protocol Π achieves (Δ, N)-independence if Π achieves N-independence under every distribution in class Δ. We start by describing the input distributions for CR-Independence in next section.

4.4.1 Distributions for CR-Independence

Computationally independent distributions: Let $\mathcal{X} = \{\mathcal{X}^{(k)}\}_{k \in \mathbf{N}}$ be a distribution ensemble such that every distribution $\mathcal{X}^{(k)}$ is the product of n arbitrary but independent distributions $X_1, \ldots, X_n$ over $\{0,1\}$, that is, $\mathcal{X}^{(k)} = X_1 \times X_2 \times \cdots \times X_n$. Ensembles with distributions of this form are called *independent*. Let $\Phi_n = \{\mathcal{X}^{\langle \ell \rangle}\}_{\ell \in \mathbb{D}}$ be the class of all independent n-dimensional ensembles, indexed by some (possibly uncountable) set $\mathbb{D}$. Let $\Psi_{C,n}$ be the class that contains all distributions ensembles computationally close to some distribution ensemble in Φ_n, that is, for each $\mathcal{D} \in \Psi_{C,n}$ there exist a distribution ensemble $\mathcal{X}$ in Φ_n such that $\mathcal{D}$ is computationally close to $\mathcal{X}$. If $\mathcal{D} \in \Psi_{C,n}$ we say $\mathcal{D}$ is a *computationally independent* distribution ensemble. Note that the ensembles for the uniform and all singleton distributions are indeed independent.

Achieving CR-Independence: It is possible to show that, if the input distributions are computationally independent then CR-independence can be achieved. The proof of this result is postponed until Section 4.5.2.

Claim 4.4.1 Under *the assumption that enhanced trapdoor permutations exist* (cf. [Gol01, Sec. C.1]), there exists a protocol that achieves $(\Psi_{C,n}, \mathsf{CR})$-independence. ▮

Conversely, unless the input distribution $\mathcal{D}$ is computationally independent, no protocol can achieve independence according to Definition 4.3.3.

Lemma 4.4.2 Let Π be any multisender broadcast protocol and let $\mathcal{D} \notin \Psi_{C,n}$ be an input distribution ensemble. Then, Π does not achieve CR-independence under input distribution $\mathcal{D}$. ▮

Proof of Lemma 4.4.2: The proof uses the distinguisher that comes from $\mathcal{D}$ not being computationally independent to build a polynomial-time computable relation R

and an adversary A that breaks the CR-independence of any protocol which runs on input distribution $\mathcal{D}$. Details follow.

Suppose $n \geq 2$ (otherwise the result holds vacuously). Let $\mathcal{D}$ be the input distribution ensemble not in $\Psi_{C,n}$ and, for some arbitrary index $i \in [n]$, consider the induced ensembles $\mathcal{D}_i$ and $\mathcal{D}_{\overline{\{i\}}} \stackrel{\text{def}}{=} \mathcal{D}_{[n]\setminus\{i\}}$. (Observe that all distributions over 1-bit are independent and $\mathcal{D}_i$ must be in Φ_1.) For distribution $\mathcal{D}_{\overline{\{i\}}}$ there are two cases depending on whether $\mathcal{D}_{\overline{\{i\}}} \in \Psi_{C,n-1}$ or not.

Let's analyze the first case, where $\mathcal{D}_{\overline{\{i\}}} \in \Psi_{C,n-1}$. Since $\mathcal{D} \notin \Psi_{C,n}$, we know $\mathcal{D}$ is not computationally close to any distribution in Φ_n. This means that, for every distribution $\mathcal{X} \in \Phi$, there exists a probabilistic polynomial-time adversary T and a constant $c > 0$ such that (w.l.o.g.) $\Pr\left[T(\mathcal{D}^{(k)}) = 1\right] - \Pr\left[T(\mathcal{X}^{(k)}) = 1\right] > k^{-c}$ for infinitely many values of the security parameter k. Take $\mathcal{X} = \mathcal{X}' \sqcup \mathcal{D}_i$ where $\mathcal{X}'$ is the ensemble in Φ_{n-1} which is computationally close to $\mathcal{D}_{\overline{\{i\}}}$. We want to prove that, in this case, Π cannot be CR-independent under input distribution $\mathcal{D}$: There must exist an adversary A, a polynomial-time predicate R, and a bit b such that $\Big|\Pr[W_i = b] \cdot \Pr\Big[R(\mathbf{W}_{\overline{\{i\}}})\Big] - \Pr\Big[W_i = b \wedge R(\mathbf{W}_{\overline{\{i\}}})\Big]\Big|$ is non-negligible. Indeed, it suffices to consider the trivial adversary A, which corrupts no party, and the predicate R defined by the execution of adversary T under the randomness that gives the "best" distinguishing advantage. Fix some arbitrary party $i \in [n]$, bit $b \in \{0,1\}$ and security parameter k. In what follows, we write T_r to denote running machine T with randomness fixed to the (sufficiently long) string $r \in \{0,1\}^*$. Then, for all bit b, for all $Z_{\overline{\{i\}}} \in \{0,1\}^{n-1}$, we define a predicate R on $Z_{\overline{\{i\}}}$ as $R^b(Z_{\overline{\{i\}}}) \stackrel{\text{def}}{=} T_r(Z_{\overline{\{i\}}} \sqcup b)$, where r is the value of τ that maximizes the difference $\Pr\left[T_\tau(\mathcal{D}^{(k)}) = 1\right] - \Pr\left[T_\tau(\mathcal{X}^{(k)}) = 1\right]$.

In what follows, we prove that, independently of protocol Π, there is a bit b such that adversary A and relation R^b violate the CR-independence of Π under input distribution $\mathcal{D}$. For simplicity, let D and X denote the random variables corresponding to to distributions $\mathcal{D}^{(k)}_{\overline{\{i\}}}$ and $\mathcal{X}^{(k)}_{\overline{\{i\}}}$ respectively. Then, for $b \in \{0,1\}$, we have

$$\begin{aligned} p_b &\stackrel{\text{def}}{=} \Pr\Big[W_i = b \wedge R^b(W_{\overline{\{i\}}})\Big] - \Pr[W_i = b] \cdot \Pr\Big[R^b(W_{\overline{\{i\}}})\Big] \\ &= \Pr[T_r(D \sqcup b) \mid \mathcal{D}_i = b] \cdot \Pr[\mathcal{D}_i = b] - \Pr[\mathcal{D}_i = b] \Pr[T_r(D \sqcup b)] \end{aligned}$$

and, therefore

$$\begin{aligned} p_0 + p_1 &= \Pr[T_r(\mathcal{D})] - (\Pr[D_i = 0] \cdot \Pr[T_r(D \sqcup 0)] \\ &\quad + \Pr[D_i = 1] \cdot \Pr[T_r(D \sqcup 1)]) \end{aligned} \tag{4.5}$$

Now, let us consider the probability that T outputs 1 under distribution $\mathcal{X} \in \Phi_n$. Since $\mathcal{X}$ is the product of independent distributions

$$\Pr[T_r(\mathcal{X})] = \Pr[X_i = 0] \cdot \Pr[T_r(X \sqcup 0)] + \Pr[X_i = 1] \cdot \Pr[T_r(X \sqcup 1)] \tag{4.6}$$

Subtracting Equation (4.6) from Equation (4.5), and using that $\mathcal{X}_i = \mathcal{D}_i$ and triangular inequality we obtain

$$\begin{aligned} |p_0| + |p_1| &\geq k^{-c} + \Pr[\mathcal{D}_i = 0] \cdot (\Pr[T_r(X \sqcup 0)] - \Pr[T_r(D \sqcup 0)]) \\ &\quad + \Pr[\mathcal{D}_i = 1] \cdot (\Pr[T_r(X \sqcup 1)] - \Pr[T_r(D \sqcup 1)]) \quad \geq \quad k^{-c'}. \end{aligned}$$

for some constant $c' > 0$. Notice that the last inequality above follows from $\mathcal{D}_{\overline{\{i\}}} \in \Psi_{C,n-1}$.

For the second case, $\mathcal{D}_{\overline{\{i\}}} \notin \Psi_{C,n-1}$, we can apply the above argument on distribution $\mathcal{D}_{\overline{\{i\}}}$ instead, and inductively on n if necessary. This is possible if we pick i so the resulting distribution $\mathcal{D}_{\overline{\{i\}}}$ is *not* computationally independent. (If that is not possible, the first case above applies and the result holds.) The base case occurs when a distribution of the form $\mathcal{D}_{j_1} \times \mathcal{D}_{j_2}$ is reached. Then, $\mathcal{D}_{j_1}$ is clearly computationally close to itself and the first case above applies. This concludes the proof. ∎

4.4.2 Distributions for G-Independence

Locally Independent distributions: We say distribution ensemble $\mathcal{D}$ is *locally independent* if for all subset $B \subset [n]$, all string $\mathbf{u} \in \{0,1\}^{|B|}$, and all string $\mathbf{w} \in \{0,1\}^{n-|B|}$ that occurs with non-zero probability as $\mathcal{D}_{\overline{B}}$, the quantity

$$\left| \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \mid \mathcal{D}_{\overline{B}}^{(k)} = \mathbf{w} \right] - \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \right] \right|$$

is negligible in the security parameter k. We denote by $\Psi_{L,n}$ the class of all locally independent distribution ensembles.

ACHIEVING G-INDEPENDENCE: We prove that G-independence can be achieved under locally independent inputs. Again, the proof of this result is postponed until Section 4.5.2.

Claim 4.4.3 Under *the assumption that enhanced trapdoor permutations exist* (cf. [Gol01, Sec. C.1]), there exists a protocol that achieves $(\Psi_{L,n}, \mathsf{G})$-independence. ▮

On the other hand, the following result shows that no protocol is G-independent under input distributions which are not locally independent.

Lemma 4.4.4 Let Π be any multisender broadcast protocol and $\mathcal{D} \notin \Psi_{L,n}$ be an input distribution. Then, Π does not achieve G-independence under input distribution $\mathcal{D}$. ▮

Proof of Lemma 4.4.4: Fix a multisender broadcast protocol Π. Let $\mathcal{D}$ be a distribution not in $\Psi_{L,n}$. Then, by definition, there exist a subset $B \subset [n]$, $|B| = t$, a string $\mathbf{u} \in \{0,1\}^t$, and a string $\mathbf{w} \in \{0,1\}^{n-t}$, such that

$$\left| \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \right] - \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \mid \mathcal{D}_{\overline{B}}^{(k)} = \mathbf{w} \right] \right|$$

is not negligible. To prove the result it suffices to choose an arbitrary index $i \in B$, and an adversary A that works as follows: A corrupts parties in B and announces 1 for (corrupted) party P_i only when $\mathbf{x}_B = \mathbf{u}$. Also, with overwhelming probability, since $\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)}$ we know that $\mathbf{W}_{\overline{B}} = \text{ANNOUNCED}_A^{\Pi}(\mathbf{x})_{\overline{B}}$ follows distribution $\mathcal{D}_{\overline{B}}^{(k)}$. Then,

$$\begin{aligned}
&\Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \right] - \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \mid \mathcal{D}_{\overline{B}}^{(k)} = \mathbf{w} \right] \\
&\quad = \Pr\left[\mathbf{W}_i = 1 \right] - \Pr\left[\mathbf{W}_i = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{w} \right] \\
&\quad = \sum_{v \in \{0,1\}^{n-t}} \left(\Pr\left[\mathbf{W}_i = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{v} \right] \cdot \Pr\left[\mathbf{W}_{\overline{B}} = v \right] \right) - \Pr\left[\mathbf{W}_i = \mathbf{u} \mid \mathbf{W}_{\overline{B}} = \mathbf{r} \right] \\
&\quad \leq \Pr\left[\mathbf{W}_i = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{v}^* \right] - \Pr\left[\mathbf{W}_i = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{r} \right]
\end{aligned}$$

is not negligible either when $\mathbf{v}^* \in \{0,1\}^{n-t}$ is the value $\mathbf{v}$ that maximizes the conditional probability $\Pr\left[\mathcal{D}_B^{(k)} = \mathbf{u} \mid \mathcal{D}_{\overline{B}}^{(k)} = \mathbf{v} \right]$. The symmetric case follows from taking $\mathbf{v}^*$ to be the value that mimimizes the probability. The result of the lemma then follows. ▮

4.4.3 Distributions for Sb-Independence

In this section, we show that Sb-Independence can be achieved under any input distribution. We first notice that Sb-Independence under class Singleton and (All, Sb)-Independence are equivalent (this follows from Proposition A.1.1). Then, we recall the results by Yao and (independently) by Goldreich et al. [Yao86, GMW87] which present protocols that securely implement any function. In particular, these protocols securely implement f_{SB}. By observing that such protocols work for any fixed input, we then have

Corollary 4.4.5 [Yao86, GMW87] Under *the assumption that enhanced trapdoor permutations exist* (cf. [Gol01, Sec. C.1]), there exists a protocol that achieves Sb-independence for any input distribution. ▌

4.4.4 Relations between Distributions

We introduce some notation first. Let Singleton be the class of all singleton input distribution ensembles. That is, for each string $\alpha \in \{0,1\}^n$, the distribution $\mathcal{D}_\alpha = \{\mathcal{D}_\alpha^{(k)}\}_{k\in\mathbf{N}}$ is in Singleton if for every k, $\mathcal{D}_\alpha^{(k)}$ assigns probability one to the string α. Let Uniform be the class whose only element is the uniform distribution ensemble, and let All be the class of all input distribution ensembles over n-bit strings. For notational convenience, in the rest of the chapter, we denote by $\mathcal{D}(\mathsf{N})$ the class of distributions associated to definition N, that is, $\mathcal{D}(\mathsf{CR}) \stackrel{\text{def}}{=} \Phi_{C,n}$, $\mathcal{D}(\mathsf{G}) \stackrel{\text{def}}{=} \Phi_{L,n}$, and $\mathcal{D}(\mathsf{Sb}) \stackrel{\text{def}}{=} \mathsf{All}$.

The following claim shows that the input distributions under which G, CR, and Sb are achievable are strictly contained in the same order. All classes also contain the class of all singleton distributions and the class of the uniform distribution. The proofs are easy and therefore omitted.

Claim 4.4.6 $\mathsf{Singleton}, \mathsf{Uniform} \subsetneq \mathcal{D}(\mathsf{G}) \subsetneq \mathcal{D}(\mathsf{CR}) \subsetneq \mathcal{D}(\mathsf{Sb})$. ▌

4.5 Implications and Separations

In this section, we compare the definitions of independence of [CR87, Gen00] with the simulation-based definition. We say a distribution is *trivial* for notion N if

every protocol achieves N-independence under that input distribution. Also, a class is trivial for notion N if every protocol achieves N-Independence under all distributions in the class. Our first implication shows that any protocol that achieves Sb-Independence must achieve CR-Independence for all achievable distributions.

Lemma 4.5.1 For every protocol Π, if Π achieves $(\mathcal{D}(\mathsf{CR}), \mathsf{Sb})$-Independence then Π also achieves $(\mathcal{D}(\mathsf{CR}), \mathsf{CR})$-Independence. ∎

Proof of Lemma 4.5.1: Assume multisender broadcast protocol Π is not CR-independent for some input distribution $\mathcal{D} \in \mathcal{D}(\mathsf{CR})$. Then, there exists an adversary A, honest party P_ℓ, and a polynomial-time predicate R such that the quantity defined in Definition 4.3.3 is not negligible under input distribution $\mathcal{D}$. We show how to transform A, R and $\mathcal{D}$, into an adversary A', and an algorithm T such that Π is not $(\mathcal{D}(\mathsf{CR}), \mathsf{Sb})$-independent. Details follow.

First, since Π is not CR-independent under input distribution $\mathcal{D} \in \mathcal{D}(\mathsf{CR})$, there must exist an adversary A (corrupting players in $B \subset [n]$), an honest party P_i, and a polynomial-time computable predicate R such that there exists constant $c > 0$ and infinitely many values of k for which (w.l.o.g.)

$$\Pr\left[\, \mathbf{W}_\ell = 1 \ \wedge \ R(\mathbf{W}_{\overline{\{\ell\}}}) \,\right] - \Pr\left[\, \mathbf{W}_\ell = 1 \,\right] \cdot \Pr\left[\, R(\mathbf{W}_{\overline{\{\ell\}}}) \,\right] \ \geq \ k^{-c} \qquad (4.7)$$

Now, let adversary A' be identical to A. We build distinguisher T from predicate R as follows:

Distinguisher $T\,(1^k, z, \mathbf{x}, \tau)$: From transcript τ extract unique $\mathbf{W}$

Output 1 if $\Big(\mathbf{W}_\ell = 1$ and $R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\Big)$

and 0 otherwise.

Algorithm T is polynomial-time since R is so. It remains to prove that T successfully distinguishes ensembles $\text{XEXEC}^{\Pi}_{A'}$ and $\text{XEXEC}^{\mathsf{Ideal}(f_{SB})}_{S}$ (as defined in equations (4.1) and (4.2)) when the input distribution is $\mathcal{D}$.

Let S be an ideal process adversary (simulator). We denote by $S(\mathbf{x}_B; z)$ the $|B|$-dimensional vector given by simulator S to function f_{SB} (in the ideal world) as the input

corresponding to corrupted parties. String z is the auxiliary input of S. Let $\Pr_1[E]$ be the probability of event E under the case $\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)}$ and $\mathbf{W} \leftarrow \textsc{Announced}_A^{\Pi}(\mathbf{x})$, and $\Pr_0[E]$ be the probability of event E under the choice $\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)}$ and $\mathbf{W} \leftarrow \mathbf{x}_{\overline{B}} \sqcup S(\mathbf{x}_B)$. Then,[5]

$$\begin{aligned}
p_1 &\stackrel{\text{def}}{=} \Pr\left[\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)} \,:\, T(1^k, z, \mathbf{x}, \textsc{Exec}_{A'}^{\Pi}(k, z, \mathbf{x})) = 1\right] \\
&= \Pr_1\left[\mathbf{W}_\ell = 1 \wedge R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\right] \\
p_0 &\stackrel{\text{def}}{=} \Pr\left[\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)} \,:\, T(1^k, z, \mathbf{x}, \textsc{Exec}_{S}^{\mathsf{Ideal}(f_{SB})}(k, z, \mathbf{x})) = 1\right] \\
&= \Pr_0\left[\mathbf{x}_\ell = 1 \wedge R(\mathbf{x}_{\overline{B}\setminus\{\ell\}} \sqcup S(\mathbf{x}_B; z)) = 1\right]
\end{aligned}$$

At this point, we use that $\mathcal{D}$ is computationally independent. Let $\mathcal{X} \stackrel{\text{def}}{=} \mathcal{D}_{\overline{\{\ell\}}} \sqcup \mathcal{D}_\ell$. By a hybrid argument, we assume $\mathcal{X} \in \Phi_n$. Then, there exists a negligible function $\epsilon(k)$ such that $|\Pr[F(\mathcal{D}^{(k)}) = 1] - \Pr[F(\mathcal{X}^{(k)}) = 1]| < \epsilon(k)$ for any probabilistic polynomial-time distinguisher F, in particular $F(\mathbf{Z}) \stackrel{\text{def}}{=} (\mathbf{Z}_\ell = 1 \wedge R(\mathbf{Z}_{\overline{B}\setminus\{\ell\}} \sqcup S(\mathbf{Z}_B; z)) = 1)$. Therefore,

$$\begin{aligned}
p_0 &< \Pr\left[\mathbf{u} \stackrel{R}{\leftarrow} \mathcal{X}^{(k)} \,:\, \mathbf{u}_\ell = 1 \wedge R(\mathbf{u}_{\overline{B}\setminus\{\ell\}} \sqcup S(\mathbf{u}_B; z)) = 1\right] + \epsilon(k) \\
&= \Pr\left[\mathbf{u}_{\overline{\{\ell\}}} \stackrel{R}{\leftarrow} \mathcal{D}_B^{(k)} \,:\, R(\mathbf{u}_{\overline{B}\setminus\{\ell\}} \sqcup S(\mathbf{u}_B; z)) = 1 \mid \mathbf{u}_\ell = 1\right] \\
&\quad \cdot \Pr\left[\mathbf{u}_\ell \stackrel{R}{\leftarrow} \mathcal{D}_\ell^{(k)} \,:\, \mathbf{u}_\ell = 1\right] + \epsilon(k) \\
&= \Pr_0\left[R(\mathbf{x}_{\overline{B}\setminus\{\ell\}} \sqcup S(\mathbf{x}_B; z)) = 1\right] \cdot \Pr_0[\mathbf{W}_\ell = 1] + \epsilon(k) \\
&< \Pr_1\left[R(\mathbf{W}_{\overline{\{\ell\}}})\right] \cdot \Pr_0[\mathbf{W}_\ell = 1] + \epsilon(k)
\end{aligned}$$

We justify last inequality as follows: (a) if $\Pr_1\left[R(\mathbf{W}_{\overline{\{\ell\}}})\right] < \Pr_0\left[R(\mathbf{W}_{\overline{\{\ell\}}})\right]$ then it suffices to consider the negated predicate $\overline{R}$ instead of R, and (b) any adversary A cannot use the simulator S's strategy otherwise A would contradict Equation (4.7) since $\mathcal{D} \in \mathcal{D}(\mathsf{CR})$. Also, by the correctness of Π, $\mathbf{W}_i = \mathbf{x}_i$ for all honest $i \in \overline{B}$. Combining the above equations with Equation (4.7), we obtain

$$\begin{aligned}
p_1 - p_0 &> \Pr_1\left[\mathbf{W}_\ell = 1 \wedge R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\right] - \Pr_1\left[R(\mathbf{W}_{\overline{\{\ell\}}})\right] \cdot \Pr_1[\mathbf{W}_\ell = 1] - \epsilon(k) \\
&> k^{-c'}
\end{aligned}$$

[5] In the rest of the proofs in this chapter, for simplicity, we assume that $\mathbf{W}_\ell = \mathbf{x}_\ell$ with probability one for all uncorrupted P_ℓ. The cases when the equality holds with overwhelming probability are analogous, although slightly more involved.

for some constant $c' > 0$ and infinitely many values of k. ▌

Similarly, all protocols that achieve CR-Independence under all distributions for which G-Independence is achievable must indeed achieve G-Independence under the same class.

Lemma 4.5.2 For every protocol Π, if Π achieves $(\mathcal{D}(\mathsf{G}), \mathsf{CR})$-Independence then Π also achieves $(\mathcal{D}(\mathsf{G}), \mathsf{G})$-Independence. ▌

Proof of Lemma 4.5.2: Let Π be a multisender broadcast protocol. Assume Π is not G-Independent under some distribution $\mathcal{D}$. We want to prove that there exist a distribution $\mathcal{D}'$ under which Π is not CR-Independent. By Proposition A.1.7, if Π does not achieve G-Independence under distribution $\mathcal{D}$, then Π is not G^{**}-Independent. Therefore, there exists an polynomial-time adversary A corrupting set $B \subset [n]$, a string $z \in \{0,1\}^*$, $i \in B$, and vectors $\mathbf{w} \in \{0,1\}^B$, $\mathbf{r}, \mathbf{s} \in \{0,1\}^{\overline{B}}$ such that the quantity

$$\Big| \Pr\Big[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{r}) \,:\, \mathbf{W}_i = 1 \Big] - \Pr\Big[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{s}) \,:\, \mathbf{W}_i = 1 \Big] \Big|$$

is not negligible. By a hybrid argument, we can assume $\mathbf{r}$ and $\mathbf{s}$ differ on a single bit, the ℓ-th one, so $\mathbf{r}_{B\setminus\{\ell\}} = \mathbf{s}_{B\setminus\{\ell\}}$. W.l.o.g. $\mathbf{r}_\ell = 0$ and $\mathbf{s}_\ell = 1$.

We build a new adversary A' identical to A and fix the honest player P_ℓ. We also define the predicate $R(\mathbf{Z}_{\overline{\ell}}) \stackrel{\text{def}}{=} (\mathbf{Z}_i \stackrel{?}{=} 1)$. Now, consider the distribution $\mathcal{D}'$ that assigns some non-negligible probability p_ℓ to the event $\mathcal{D}'^{(k)}_\ell = 1$, and probability one to $\mathcal{D}'^{(k)}_{\overline{\{\ell\}}} = (\mathbf{w} \sqcup \mathbf{r}_{\overline{B}\setminus\{\ell\}})$. Notice that $\mathcal{D}'^{(k)}$ is in $\mathcal{D}(\mathsf{G})$ but it is not trivial. Let $\Pr_{\mathcal{D}'}[E]$ the probability of event E when $\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathcal{D}'^{(k)})$. Since P_ℓ is honest $\Pr_{\mathcal{D}'}[\mathbf{W}_\ell = 1] = \Pr\Big[\mathcal{D}'^{(k)}_\ell = 1\Big] = p_\ell$. Then,

$$\begin{aligned}\Pr_{\mathcal{D}'}\Big[R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\Big] &= (1-p_\ell)\cdot \Pr\Big[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{r}) \,:\, \mathbf{W}_i = 1\Big] \\ &\quad + p_\ell \cdot \Pr\Big[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{s}) \,:\, \mathbf{W}_i = 1\Big]\end{aligned}$$

and

$$\begin{aligned}\Pr_{\mathcal{D}'}\Big[\mathbf{W}_\ell = 1 \wedge R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\Big] &= \Pr_{\mathcal{D}'}[\mathbf{W}_\ell = 1 \wedge \mathbf{W}_i = 1] \\ &= p_\ell \cdot \Pr\Big[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{s}) \,:\, \mathbf{W}_i = 1\Big]\end{aligned}$$

Putting it all together,

$$\begin{aligned}
&\left|\Pr_{\mathcal{D}'}\left[\,\mathbf{W}_\ell = 1\,\right]\cdot\Pr_{\mathcal{D}'}\left[\,R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\,\right] - \Pr_{\mathcal{D}'}\left[\,\mathbf{W}_\ell = 1 \wedge R(\mathbf{W}_{\overline{\{\ell\}}}) = 1\,\right]\right| \\
&= \quad p_\ell\cdot(1-p_\ell)\cdot\Big|\Pr\left[\,\mathbf{W}\leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w}\sqcup\mathbf{r})\,:\,\mathbf{W}_i = 1\,\right] \\
&\qquad - \Pr\left[\,\mathbf{W}\leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w}\sqcup\mathbf{s})\,:\,\mathbf{W}_i = 1\,\right]\Big|
\end{aligned}$$

which is not negligible. ∎

4.5.1 Separations

At this point, we look into whether the definitions are equivalent when restricted to achievable input distributions. Proposition 4.5.3 shows this is not the case. We prove that there are distributions for which the definition of [CR87] always holds no matter the protocol, but that this cannot happen with Sb-Independence.

Proposition 4.5.3 The class Singleton is trivial for CR independence but not trivial for Sb independence. ∎

Proof of Proposition 4.5.3: The proof follows easily from the definition of CR independence – under a fixed input all probabilities collapse to either 0 or 1 with overwhelming probability, for any protocol. Then, consider a protocol that does not achieve Sb-Independence (we know such protocol exist). Since (Singleton, Sb)-Independence is equivalent to Sb-Independence, it follows that such protocol cannot achieve Sb-Independence under class Singleton. ∎

It is also possible to show that the definitions of [CR87] and [Gen00] are not equivalent, but instead that G-independence is strictly weaker than CR-independence.

Lemma 4.5.4 There exists a protocol Π_G which achieves $(\mathcal{D}(\mathsf{G}), \mathsf{G})$-independence but does not achieve CR-independence for any input distribution in $\mathcal{D}(\mathsf{G})$. In particular, Π_G is G-Independent for the uniform distribution, but not CR-Independent for the uniform distribution. ∎

Proof: We show a protocol implementing multisender broadcast that, even though it satisfies Definition 4.3.4 (i.e., the notion of simultaneous broadcast of [Gen00]), it violates Definition 4.3.3 (i.e., the definition of independence of [CR87]). The "flawed" protocol Π_G uses a subprotocol Θ which essentially performs a simultaneous broadcast *unless* two corrupted parties misbehave in a very controlled manner – by setting some auxiliary input bit to 1. In such case, protocol Θ reveals some information about the honest parties' inputs to two corrupted parties. The leakage of information is done in such a way that the output of each single corrupted party is *not* correlated to the outputs of honest parties, but the *combined* outputs are.

We describe protocol Θ first. Protocol Θ is a n-party protocol that securely implements function $g(\mathbf{v})$ on input $\mathbf{v} = (v_1, \ldots, v_n)$ defined as

$$g(\mathbf{v}) \stackrel{\text{def}}{=} \begin{cases} \text{First, parse each } v_i \text{ as } (x_i, b_i) \\ \text{Pick } r \stackrel{R}{\leftarrow} \{0,1\} \text{ and set } \mathcal{L} \leftarrow \{\, i \,:\, b_i = 1 \,\} \\ \text{If } |\mathcal{L}| = 2 \text{ then set } \ell_1, \ell_2 \in \mathcal{L},\ \ell_1 < \ell_2, \text{ otherwise set } \ell_1, \ell_2 \leftarrow 0 \\ \text{Compute } y \stackrel{R}{\leftarrow} \bigoplus_{i \notin \{\ell_1, \ell_2\}} x_i \\ \text{Set } w_i \leftarrow \begin{cases} r & \text{if } |\mathcal{L}| = 2 \text{ and } i = \ell_1 \\ r \oplus y & \text{if } |\mathcal{L}| = 2 \text{ and } i = \ell_2 \\ x_i & \text{if } i \neq \ell_1, \ell_2 \end{cases} \\ \text{Set } \mathbf{w} \leftarrow (w_1, \ldots, w_n) \text{ and output the } n\text{-dimensional vector } (\mathbf{w}, \mathbf{w}, \ldots, \mathbf{w}) \end{cases}$$

For simplicity, we write the input vector $\mathbf{v}$ as $\mathbf{v} = (\mathbf{x}, \mathbf{b})$, where $\mathbf{x}, \mathbf{b} \in \{0,1\}^n$. We first notice that a protocol that securely implements function g can be built using known techniques (cf. [BGW88, GMW87, CCD88]) as long as $t < \lceil n/2 \rceil$.

Claim 4.5.5 There exist a protocol Θ that securely implements function g (in the sense of [Can00a]). ▮

We now describe protocol Π_G. On private input $x_i \in \{0,1\}$, each party P_i sets up an auxiliary bit $b_i \leftarrow 0$. Then, all parties call subprotocol Θ on input $((x_1, b_1), (x_2, b_2), \ldots, (x_n, b_n))$. Let $\mathbf{W}_i$ be the vector obtained as the output of protocol Θ by party P_i. Each party P_i outputs $\mathbf{W}_i$ as the final protocol result.

We show that protocol Π_G is *not* CR-Independent under any non-trivial input distribution. Indeed, there exists an adversary A^* such that, when protocol Π_G is executed on any input $\mathbf{x}$ under adversary A^*, the sum (mod 2) of the announced bits is always zero. Adversary A^* corrupts only two parties and instructs them to set their auxiliary bits to 1. The next claim follows directly from the definition of g.

Claim 4.5.6 Assume parties have inputs chosen according to some arbitrary distribution $\mathcal{D} \in \mathcal{D}(\mathsf{G})$. There exists an adversary A^* such that the execution of protocol Π_G on input $\mathbf{x} \in \mathcal{D}$ under adversary A^* defines a vector of announced bits $\mathbf{W}$ satisfying $\bigoplus_i W_i = 0$. ∎

The attack works for any non-trivial distribution, i.e., any distribution that is not statistically close to a singleton. For any such distribution, there must exists and index i such that $1/poly < \Pr[\,\mathbf{W}_i = 0\,] < 1 - 1/poly$. The above claim gives an adversary and a polynomial-time predicate we can use to correlate the output of the corrupted parties with the output of an honest party P_i, namely $R(\mathbf{Z}_{\overline{\{i\}}}) \stackrel{\text{def}}{=} (\oplus_{j \neq i} \mathbf{Z}_j = 0)$. Notice that the predicate holds if and only if P_i announces 0.

We now show that protocol Π_G achieves G-Independence for any non-trivial, locally independent input distribution $\mathcal{D}$. Indeed, for any adversary A that succeeds on attacking the G-Independence of Π_G under $\mathcal{D}$, we exhibit a distinguisher Q that contradicts the security of Θ (Claim 4.5.5). We proceed as follows. Assume Π_G is not G-Independent. By Proposition A.1.7, Π_G is not G^{**}-Independent. Then there is an adversary A which corrupts parties in B, an auxiliary input τ, and a corrupted party P_i, for which there are vectors $\mathbf{w} \in \{0,1\}^B$, $\mathbf{r}, \mathbf{s} \in \{0,1\}^{\overline{B}}$, such that

$$\Big|\Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi_G}_{A(k,\tau)}(\mathbf{w} \sqcup \mathbf{r}) \;:\; \mathbf{W}_i = 1\,\Big] - \Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi_G}_{A(k,\tau)}(\mathbf{w} \sqcup \mathbf{s}) \;:\; \mathbf{W}_i = 1\,\Big]\Big|$$

is not negligible. By a hybrid argument, we can assume $\mathbf{r}$ and $\mathbf{s}$ differ in a single bit, the ℓ-bit, so $r_\ell \neq s_\ell$, and w.l.o.g, $r_\ell = 0$ and $s_\ell = 1$.

The above adversary gives us a procedure to guess the input bit used by honest party P_ℓ in protocol Θ as long as the inputs vector for the remaining parties is equal to $\mathbf{w} \sqcup \mathbf{r}_{\overline{B} \setminus \{\ell\}}$.

Indeed, starting from A we show how to build an adversary A' for Θ, such that for any ideal-process adversary S for $\mathsf{Ideal}(g)$, there exist a distinguisher Q, an auxiliary input z', an input vector $\mathbf{v}' = (\mathbf{x}', \mathbf{b}')$, such that the quantity

$$\left|\Pr\left[Q(1^k, z', \mathbf{v}', \textsc{Exec}^{\Theta}_{A'}(k, z', \mathbf{v}')) = 1\right] - \Pr\left[Q(1^k, z', \mathbf{v}', \textsc{Exec}^{\mathsf{Ideal}(g)}_{S}(k, z', \mathbf{v}')) = 1\right]\right|$$

is not negligible. Adversary A' is simple. It corrupts the same parties as B, and works as follows. On input $(\mathbf{x}_B, \mathbf{b}_B)$, A' simply discards $\mathbf{b}_B$ and then simulates A. We now set $\mathbf{b}' = \mathbf{0}$ and $z' = \tau$. For simplicity, for any vector $\mathbf{x} \in \{0,1\}^n$, denote

$$\begin{aligned} q_{real,\mathbf{x}} &\stackrel{\text{def}}{=} \Pr\left[Q(1^k, z', (\mathbf{x}, \mathbf{b}'), \textsc{Exec}^{\Theta}_{A'}(k, z', (\mathbf{x}, \mathbf{b}'))) = 1\right], \\ q_{ideal,\mathbf{x}} &\stackrel{\text{def}}{=} \Pr\left[Q(1^k, z', (\mathbf{x}, \mathbf{b}'), \textsc{Exec}^{\mathsf{Ideal}(g)}_{S}(k, z', (\mathbf{x}, \mathbf{b}'))) = 1\right] \end{aligned}$$

It remains to show a distinguisher algorithm Q that works with good probability. Our algorithm Q takes as input a security parameter $k \in \mathbf{N}$, an auxiliary string $z \in \{0,1\}^*$, a vector $\mathbf{v} = (\mathbf{x}, \mathbf{b}) \in \{0,1\}^n \times \{0,1\}^n$, and a string Z drawn either from distribution $\textsc{Exec}^{\Theta}_{A'}(k, z, \mathbf{v})$ or distribution $\textsc{Exec}^{\mathsf{Ideal}(g)}_{S}(k, z, \mathbf{v})$. Thus, on input $(1^k, z, (\mathbf{x}, \mathbf{b}), Z)$, algorithm Q first extract the corrupted set B and the unique vector $\mathbf{W} = (W_1, \ldots, W_n)$ of announced values from transcript Z. Then, it simply outputs 1 if $(W_i = W_\ell)$, and 0 otherwise. Let $\mathbf{x}^{\mathbf{r}} = \mathbf{w} \sqcup \mathbf{r}$ and $\mathbf{x}^{\mathbf{s}} = \mathbf{w} \sqcup \mathbf{s}$. By definition of distinguisher Q and adversary A', in the real model we have that

$$\begin{aligned} q_{real,\mathbf{x}^{\mathbf{s}}} &= \Pr\left[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi_G}_{A(k,\tau)}(\mathbf{x}^{\mathbf{s}}) \,:\, \mathbf{W}_i = 1\right] \\ q_{real,\mathbf{x}^{\mathbf{r}}} &= \Pr\left[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi_G}_{A(k,\tau)}(\mathbf{x}^{\mathbf{r}}) \,:\, \mathbf{W}_i = 0\right] \end{aligned}$$

In the ideal model, on the other hand, the adversary S has access only to $\mathbf{x}_B = \mathbf{w}$, and therefore

$$q_{ideal,\mathbf{x}^{\mathbf{r}}} = 1 - \Pr\left[S(\mathbf{w}; \tau)_i = 1\right] \quad \text{and} \quad q_{ideal,\mathbf{x}^{\mathbf{s}}} = \Pr\left[S(\mathbf{w}; \tau)_i = 1\right]$$

Combining the above equations, we obtain

$$\begin{aligned} |q_{real,\mathbf{x}^{\mathbf{s}}} - q_{ideal,\mathbf{x}^{\mathbf{s}}}| + |q_{real,\mathbf{x}^{\mathbf{r}}} - q_{ideal,\mathbf{x}^{\mathbf{r}}}| &\geq |q_{real,\mathbf{x}^{\mathbf{s}}} - q_{real,\mathbf{x}^{\mathbf{r}}} - (q_{ideal,\mathbf{x}^{\mathbf{r}}} + q_{ideal,\mathbf{x}^{\mathbf{s}}})| \\ &= \Big|\Pr\left[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi_G}_{A(k,\tau)}(\mathbf{x}^{\mathbf{s}}) \,:\, \mathbf{W}_i = 1\right] \\ &\quad - \Pr\left[\mathbf{W} \leftarrow \textsc{Announced}^{\Pi_G}_{A(k,\tau)}(\mathbf{x}^{\mathbf{r}}) \,:\, \mathbf{W}_i = 1\right]\Big| \end{aligned}$$

which is not negligible by the G^{**}-Independence. Therefore, for either input $\mathbf{x}' = \mathbf{x}^{\mathbf{s}}$ or input $\mathbf{x}' = \mathbf{x}^{\mathbf{r}}$, the quantity $|q_{real,\mathbf{x}'} - q_{ideal,\mathbf{x}'}|$ is not negligible. This concludes the proof of the lemma. ∎

We remark that the previous lemma indicates that G-Independence is not only weaker than the other definitions, but also rather unsatisfactory. Indeed, by following G-Independence, we may deem protocols like Π_G "secure", when in reality they fail to provide even a very intuitive notion of independence – namely the one that requires the announced bits *do not always* sum 0. We stress the above result holds even for the uniform distribution.

4.5.2 Feasibility of CR and G independence

At this point, we have all the tools needed to prove the feasibility results for CR and G-Independence, namely that there exist protocols that achieve $(\mathcal{D}(\mathsf{CR}), \mathsf{CR})$-Independence as well as $(\mathcal{D}(\mathsf{G}), \mathsf{G})$-Independence. Indeed, Corollary 4.4.5 together with Claim 4.4.6 and the results of this section provide concise proofs for Claim 4.4.1 and Claim 4.4.3. Claim 4.4.1 follows from the existence of a protocol achieving $(\mathcal{D}(\mathsf{Sb}), \mathsf{Sb})$-Independence (by Corollary 4.4.5), and that $(\mathcal{D}(\mathsf{CR}), \mathsf{Sb})$-Independence implies $(\mathcal{D}(\mathsf{CR}),$ CR)-Independence. Claim 4.4.3 is proved analogously.

Reference: Most material in this chapter is a reprint from the material appearing in "Simultaneous Broadcast Revisited," Alejandro Hevia and Daniele Micciancio, in the proceedings of 24th ACM Symposium on Principles of Distributed Computing (PODC 2005), Marcos Kawazoe Aguilera, James Aspnes Eds., ACM Press, 2005.

5

Universally Composable Simultaneous Broadcast

5.1 Introduction

5.1.1 The Need for Efficient Simultaneous Broadcast with Strong Security Guarantees

As mentioned in Chapter 4, the concept of *simultaneous broadcast* was first introduced by Chor et al. in [CGMA85], along with a simulation-based definition. In terms of efficiency (round complexity), the n-party protocol presented by Chor et al. required $\mathcal{O}(n)$ rounds for each simultaneous broadcast operation. Chor and Rabin [CR87] reduced the round complexity to $\mathcal{O}(\log n)$ rounds, and finally Gennaro [Gen00], working in the common random string model, obtained a protocol with constant round complexity. Our results in the previous chapter showed that, even as the round-efficiency of the solutions increased, the definitions of security did not remain the same, and they actually became increasingly restricted. In particular, the protocol presented in [Gen00], the most round efficient protocol so far, is secure under a definition of security strictly weaker than the original simulation-based definition of [CGMA85].[1] Nonetheless, the round efficiency

[1] Indeed, the definition of simultaneous broadcast proposed in [Gen00] may not exclude protocols that fail to achieve the intuitive notion of independence captured by the simulation-based definition of

of Gennaro's protocol made the search attractive for either a proof that such protocol achieves a stronger notion of simultaneous broadcast (eg. [CGMA85]), or for a variant of that protocol that does it.

Concurrent Execution and Universal Composability: Until recently, most cryptographic primitives were seen as stand-alone protocols and analyzed as such. The development of increasingly complex computing environments, brought the concern that previously secure protocols (proven as stand-alone primitives) might not remain secure under stronger adversarial conditions, like parallel or concurrent execution of many (possibly different) protocols [GK96, DDN01], or if invoked by other (possibly unknown) protocols. It was in such context that several security frameworks were developed (see [MR91, BCG93, Can00a, HM00, DM00, PSW00, PW00, PW01, Can01b, Can05]). Among those, in [Can01b, Can05], Canetti presented the *Universally Composable (UC) Security* framework, which allows modular description and analysis of protocols under concurrent execution and provides strong composability guarantees. Very informally, in the UC framework, security of a protocol is formalized in terms of an ideal object, a functionality, which acts as a reactive trusted party and performs computations on behalf of the users. A protocol is secure if it is a good "replacement" of a given functionality, where the quality of replacement is measured in terms of how well an execution of the protocol against a realistic adversary "emulates" an ideal process in which the computation is done by the functionality (see [Can05] for details). Indeed, UC secure protocols remain so under general composition with an unbounded number of instances of arbitrary protocols running concurrently. Thus, given the benefits of achieving UC security, the security of many cryptographic primitives has been revisited to explore whether these stronger UC guarantees can be achieved, and if so, at what cost (in terms of efficiency or assumptions). As we will see in the next section, simultaneous broadcast can be achieved under UC security not only at no extra cost, but also with *gains* in terms of efficiency.

[CGMA85].

5.1.2 New Results

In this work, we present a communication and round efficient solution for the simultaneous broadcast problem. Our solution is based on verifiable secret sharing (VSS) [CGMA85] and does not uses zero-knowledge proofs, zero-knowledge proofs of knowledge, or commitments schemes as previous constructions [CR87, Gen00]. Moreover, our construction is provably secure in the Universally Composable framework against computationally-unbounded adaptive adversaries assuming an honest majority, with negligible error probability.[2] To achieve this, we introduce a natural definition of Simultaneous Broadcast in the UC framework which implies all previous definitions. Our simultaneous broadcast construction is very efficient: we run one VSS per party in parallel. While the construction is technically simple, proving UC security present some subtleties, like dealing with rushing adversaries or parties simply "copying" someone else's sharing. We overcome some of the problems by defining a synchronous variant of verifiable secret sharing, which we call *Terminating VSS* (TVSS), and building our simultaneous broadcast protocol invoking such TVSS functionality. We then show that, when formalized as UC functionality, TVSS is intrinsically synchronous. A benefit of this approach is that our simultaneous broadcast protocol does not explicitly requires global synchronous communication since all the synchronicity is provided by the Terminating VSS functionality. Our construction and proof exemplifies the approach, first suggested by Canetti in [Can05], of abstracting synchronous communication as a functionality rather than embedding it in the execution model [Nie03, HMq04]. We believe this approach leads to modular analysis, simple protocol design and simpler proofs.

5.1.3 Discussion and Related Work

Simultaneous Broadcast: As mentioned above, the simultaneous broadcast problem was put forward by Chor et al. [CGMA85] who proposed a simulation-based definition and a linear-round protocol. This protocol essentially executed n sequential VSS protocols, where n is the number of communicating parties. The sequential execution was

[2] Most of the properties of our solution – namely communication and round complexity, reliability and negligible error probability – are inherited from the VSS used as building block [CDD+99].

needed to prevent corrupted parties from broadcasting the same value as an honest party, for instance by reusing (copying) the VSS data sent out by the honest party. Then, Chor and Rabin in [CR87] showed how to reduce the round complexity to $\mathcal{O}(log(n))$ rounds. Their protocol requires, among other things, that each party first broadcast a commitment of her input and then proves knowledge of the broadcasted value. The reduction in rounds comes from using a clever scheduling technique for doing the proofs – for any two players, there is a step in the protocol where one player acts as prover and the other one acts as verifier of the proof of knowledge. Such a scheduling prevents "copying" the proofs. Finally, Gennaro in [Gen00] put forward a protocol that greatly simplifies the one in [CR87] by showing how to run the proofs of knowledge in parallel, essentially employing non-interactive proofs of knowledge [SP92].

In terms of the previous definitional work for simultaneous broadcast problem, it turns out that each result in [CGMA85, CR87, Gen00] presents a different definition. In the previous chapter we show that these definitions form a strict hierarchy when considered in terms of input distributions. There, we pointed out that the strongest definition (in a well-defined sense, see Section 4.5) is the simulation-based notion of [CGMA85], which preserves security under sequential composition. The notions in [CR87, Gen00] targeted stand-alone execution and thus provide no composition guarantees.

Verifiable Secret Sharing: The notion of Verifiable Secret Sharing (VSS) was first proposed by Chor et al. [CGMA85] inspired by the need of adding robustness to standard secret sharing (eg. Shamir's [Sha79], independently invented by Blakley [Bla79]). The problem has been extensively studied both in the synchronous setting (see for example [FM88, BGW88, GMW91, RBO89, FM97, CDD$^+$99, AF04]) and in the asynchronous setting [BCG93, BOKR94, CR93, CKPS01, CKLS02]. In the information-theoretic model with adaptive adversaries, Rabin and Ben-Or [RBO89] proposed a VSS secure under an honest majority, by allowing negligible failure probability. Subsequently, in the same model, Cramer et al. [CDD$^+$99] improved the information checking protocols of [RBO89] and presented a very efficient constant-round VSS protocol. In this thesis, we observe that Cramer et al.'s protocol achieves the stronger TVSS definition achieved in this thesis, and we use it we obtain our constant-round solution.

Related Protocols and Generic Solutions: The simultaneous broadcast problem is related to the idea of *common-coin* protocols (Feldman and Micali [FM97], and Micali and Rabin [MR90]), where several parties want to generate one or more unbiased coins, in a distributed way. Indeed, the constructions proposed in [FM97, MR90] involve executing VSS protocols in parallel, in a similar approach as ours. We remark, however, that the goals are different: while for common-coin generation it suffices that the broadcast value of a single (uncorrupted) value is not correlated to the output of corrupted parties,[3] in the simultaneous broadcast problem we seek to guarantee that every single component of the output vector of the broadcast values remains "uninfluenced" by the other values in the same vector. In addition, our construction must guarantee security under general (UC) composition.

A related, although orthogonal issue, is the problem of *simultaneous termination*, which has been studied by Lindell et al. [LLR02b]. The problem arises in the context of parallel composition of multiparty protocols in synchronous networks, where certain protocols (including broadcast protocols) may not terminate in the same round when composed in parallel, thus complicating their sequential composition with other protocols. (We note that, in [LLR02b], the term *simultaneous broadcast* is used but with a different meaning than ours, as they refer to what we call parallel broadcast.) The methods in [LLR02b] do not attempt to achieve independence (in fact, they do not) since their concern is not ensuring new functional properties of the resulting parallel protocol, but ensuring that they can be safely composed sequentially with other protocols while preserving round efficiency. Also in the context of composition of broadcast protocols, Lindell et al. discussed some pitfalls in the composition of Authenticated Byzantine Agreement [LLR02a]. They show that unless session identifiers are available, no parallel or concurrent composition of Authenticated Byzantine Agreement is secure if more than one third of the parties are faulty. In our work, we do assume the availability of broadcast channels (in the form of the broadcast functionality $\mathcal{F}_{BC}$) but we put no restrictions on how they are implemented. For example, session identifiers for the broadcast protocols could be initialized using the techniques of Barak et al. [BLR04], or by standard techniques under setup assumptions [CLOS02].

[3] In all fairness, in general the mentioned protocols do achieve more than that.

Lastly, there are powerful UC constructions for secure multiparty computation of generic protocols (cf. [CLOS02, DN03]) which could certainly be used to provide a solution to the simultaneous broadcast problem. Indeed, techniques of [CLOS02] do provide such a solution tolerating any number of corrupted parties in the common random string model. However, as done in the context of many other examples of multiparty secure computation (eg. threshold signatures, key-exchange, voting), our goal is to look for more specialized, and therefore more efficient, solutions for the simultaneous broadcast problem.

5.1.4 Comparison with previous solutions

Communication and round efficiency: In terms of efficiency, the number of rounds required in our construction is equal to the number of rounds of the terminating VSS construction we use, which is the one proposed by Cramer et al. [CDD+99]. Similarly, the computational complexity of our construction is n times that of the terminating VSS in [CDD+99]. Concretely, our solution requires $\mathcal{O}((k + \log n)n^4)$ bits of communication, and only 14 rounds (or 12 if no faults occur). If the model is extended so parties can use digital signatures, the protocol takes only 7 rounds, although security holds only against computationally-bounded adversaries. In comparison, the previous constant round solution [Gen00] uses a comparable number of rounds (seven for the VSS protocol [BGW88], plus six for the computational zero-knowledge proof [GMW91], plus three rounds) but requires the communication of a large number of bits (n copies of a non-interactive zero-knowledge proof of knowledge for a generic NP statement [SP92], plus n times the communication complexity of the VSS protocol of [GMW91], at total that in most implementations is often orders of magnitude larger than $\mathcal{O}((k+\log n)n^4)$).

Adversaries and Resilience: In terms of the tolerated adversary our solution is similar to Gennaro's. The construction of [Gen00] works over public channels under computationally-bounded static adversaries and can be made secure under adaptive adversaries using the compiler of [CFGN96]. In comparison, assuming secure channels, our solution tolerates computationally unbounded adaptive adversaries, and security in the public channel setting (and computationally-bounded adversaries) can be obtained

by standard techniques (like non-committing encryption [CFGN96, DN00]). In terms of resilience, our construction tolerates at most $t < n/2$ corrupted parties as Gennaro's solution. The constructions of Chor et al. [CGMA85] and Chor and Rabin [CR87] tolerate $t < n/4$ and $t < n/2$ respectively.

ORGANIZATION: The chapter is organized as follows. In the next section, we briefly recall our model of computation (the UC framework). Then, in Section 5.3, we describe and justify our formalization of the notion of Terminating VSS (denoted UC-TVSS), the synchronous variant of VSS considered here, and we mention how it can be efficiently implemented. Section 5.5.1 presents our notion of security for simultaneous broadcast (denoted UC-SB), and shows how to implement it from UC-TVSS. We conclude in Section 5.6 by discussing how to extend our results to the public channel model, and how to model simultaneous broadcast under purely asynchronous communication.

5.2 Preliminaries

THE UC FRAMEWORK: Our results are in the Universally Composable framework of Canetti [Can05] as described in Section 2.2.4. For convenience, we briefly outline the basic underlying ideas here. In the UC framework, the desired properties of cryptographic protocols are defined in terms of tasks or *functionalities*. A functionality is a "trusted third party" that first obtains inputs directly from the parties, performs certain instructions on these inputs, and provides the parties with the appropriate outputs. A protocol securely implements a given cryptographic task, if executing the protocol against a realistic (i.e. real-life) adversary "emulates" the execution of an *ideal process*. In the ideal process, the task is computed by the functionality directly interacting with the parties against a very limited adversary (called the *ideal-adversary*). The notion of "emulation" involves a distinguisher $\mathcal{Z}$ which, by providing the parties with inputs and seeing their outputs, and by interacting with the adversary, attempts to tell whether it is interacting with a real protocol and the real-life adversary, or with the functionality and the ideal-adversary. *Good* emulation means no such environment is successful. For more details and proofs the reader is referred to [Can05].

In this chapter, we consider a network of n parties, $P_1, \ldots, P_n$, connected with perfectly private and authenticated channels plus a broadcast channel. In the UC terminology, this translates to working in the $(\mathcal{F}_{SMT}, \mathcal{F}_{BC})$-hybrid model, where $\mathcal{F}_{SMT}$ is the *secure message transmission* functionality [Can05] and $\mathcal{F}_{BC}$ is the *broadcast channel* functionality, which does not satisfy any independence or "fairness" property (i.e. it allows *rushing*). There is a computationally unbounded adversary that can adaptively corrupt up to $t < n/2$ parties. Upon corruption, parties follow the instructions by the adversary – the adversary is active. Our protocols allow an error probability negligible in the security parameter k.

5.3 Terminating VSS (UC-TVSS)

One inconvenience of the definition of standard VSS schemes (as in Definition 2.2.1 or even in Definition 2.2.2) for our purposes is that it does not guarantee the protocol terminates if the dealer is corrupted. Nonetheless, all synchronous VSS protocols in the literature [BGW88, GMW91, RBO89, FM97, GRR98, CDD+99, AF04] seem to satisfy some form of *terminating* condition: there is a round in the execution in which all parties agree that the sharing phase has "timed-out" and the secret is fixed. To capture this property while preserving the possibility that the VSS protocol being used *from* higher-level asynchronous protocols, we define the notion of *Terminating VSS* (TVSS).

TVSS protocols are guaranteed to conclude independently of the dealer's actions. We characterize TVSS as a functionality in the UC framework, $\mathcal{F}_{TVSS}$, which is shown in Figure 5.1. Intuitively, $\mathcal{F}_{TVSS}$ extends the VSS functionality so that even if a corrupted dealer D fails to call the functionality, a fixed value is eventually associated to D. Indeed, honest parties can "force" the functionality to fix a value for D by sending `EndSharing` messages. Our formulation of $\mathcal{F}_{TVSS}$, is inspired by the UC VSS variant of Abe and Fehr (see [AF04] and Section 2.2.4), which includes the concept of "spooling" the secret, a syntactic technique that allows the dealer to announce to the adversary – via a `Spool` message – that a new functionality is being called. The adversary is thus able to adaptively corrupt the dealer before the dealer commits to a value. Another purely syntactic choice is allowing the adversary to trigger the end of the sharing phase as it is

easy to see that alternative formulations like triggering the end sharing after "enough" (say $t+1$) honest parties have sent `EndSharing` mesages are equivalent.

Definition 5.3.1 We say a protocol π achieves UC-TVSS if π UC-realizes functionality $\mathcal{F}_{TVSS}$. ∎

The name *Terminating* VSS reflects that a protocol that UC-realizes $\mathcal{F}_{TVSS}$ will conclude (terminate) as long as the adversary delivers all sent messages. The adversary can still delay or block some messages forever – but nothing more. In particular, the protocol does not stall even if the corrupted dealer is irresponsive. This adversarial behavior is a concern in protocols that depend on the termination of a VSS subprotocol, even in the authenticated transmission model, as the parties waiting for a successful completion of the VSS functionality may not have access to synchronous communication or some "time-out" mechanism.[4] In this context, *termination* means that, once honest parties are instructed to start executing the TVSS protocol, then no matter what the actions of the dealer are, π will terminate with some (possibly empty) output if the adversary delivers all the messages.

Discussion: One may argue that termination issue seem to disappear if one considers a synchronous version of the UC model (as done in [Nie03, HMq04]). While synchronous communication among all parties certainly allows to implement time-outs (and thus have default sharings if the dealer does not participate), we believe that "encapsulating" synchronicity *inside* the primitive that requires it is useful as higher-level protocols do not need to be aware of it. Concretely, TVSS not only captures a form of synchronous VSS but also keeps the dependence on synchronous communication modularized, as any reliance on it is explicitly and independently handled inside the TVSS functionality. In particular, even though it is possible to show that any implementation of TVSS requires synchronous communication (at least twice) among the parties running it,[5] a higher-level protocol ρ using the TVSS functionality can run in an asynchronous way. In practice, this means that ρ could be implemented in an asynchronous network where only lim-

[4] We remark that, in some applications, the parties "waiting" for the completion of a VSS subprotocol may not be the same as the ones executing the VSS protocol.

[5] This is implied by Claim 5.5.2 where TVSS is shown to be equivalent to functionality $\mathcal{F}_{SYN}$, synchronous communication with guaranteed delivery [Can05].

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

$\mathcal{F}_{TVSS}$ expects its SID to be of the form $sid = (sid', D, \mathcal{P})$, where $\mathcal{P}$ is a list of parties among which sharing is to be performed. It proceeds as follows.

(1) At first activation, initialize $\mathbf{s}$ as $\perp$.

(2) Upon receiving input $(\texttt{Spool}, sid, s)$ from party $D \in \mathcal{P}$, set $\mathbf{s} \leftarrow s$ and send $(\texttt{SpoolRcvd}, sid, D)$ to adversary A. (Any subsequent input `Spool` is ignored.)

(3) Upon receiving input $(\texttt{Share}, sid, s')$ from party $D \in \mathcal{P}$, set $\mathbf{s} \leftarrow s'$ and send $(\texttt{ShareRcvd}, sid, D)$ to adversary A. (Any subsequent input `Share` is ignored.)

(4) Upon receiving input $(\texttt{EndSharing}, sid)$ from uncorrupted party $P \in \mathcal{P}$, record $(\texttt{EndSharing}, P)$ and send $(\texttt{EndSharingRcvd}, sid, P)$ to adversary A.

(5) Upon receiving message $(\texttt{Corrupt}, sid, P)$ from the adversary, do: If $P = D$ then send $\mathbf{s}$ to the adversary. Otherwise, delete record $(\texttt{EndSharing}, P)$, if exists. In both cases, send $(\texttt{Corrupt}, sid)$ message to P.

(6) Upon receiving message $(\texttt{DoEndSharing}, sid)$ from the adversary do: If there is at least one record of the form $(\texttt{EndSharing}, sid, P)$, and

- D is honest and a message $(\texttt{Share}, sid, u)$ from D has been received, or
- D is corrupted,

then send $(\texttt{Shared}, sid)$ to each party in $\mathcal{P}$ and the adversary A. (Any subsequent input `Share` or message `DoEndSharing` is ignored.) Otherwise, ignore the message.

(7) Upon receiving input $(\texttt{Open}, sid)$ from uncorrupted party P, output $(\texttt{Opened}, sid, \mathbf{s})$ to P and the adversary A.

Figure 5.1 The Terminating Verifiable Secret Sharing (with Spooling) functionality, $\mathcal{F}_{TVSS}$.

ited synchronicity is available (say only within certain subsets of the parties, or when synchronous communication can only be provided very infrequently) as long as the subprotocol that realizes the TVSS functionality has "enough" access to the synchronization capability. For example, applications in cluster networks [vABH03] may exploit the advantage of implementing TVSS locally in each cluster (where synchronization is easier) while the inter-cluster protocol ρ can run asynchronously. Even our concrete application of TVSS, building simultaneous broadcast, where each TVSS involves *all* the parties, may benefit from this modular approach: dealers in different TVSS subprotocols could start the execution at different rounds (because of lack of network connectivity, for example) and still be able to achieve simultaneous broadcast. Furthermore, the simplicity of our construction for simultaneous broadcast shows that this approach may also simplify protocol design and security proofs.

5.3.1 Instantiating TVSS

In this section, we revisit the very efficient VSS protocol presented by Cramer et al. in [CDD+99] in a synchronous model of computation with some negligible error probability. The scheme is based on the bivariate solution of Feldman [FM88, BGW88] and builds on the information checking techniques of Rabin and Ben-Or [RBO89, Rab94]. The construction is very efficient: the sharing phase takes fourteen rounds and reconstruction takes two rounds,[6] while the total communication cost is $O((k + \log n)n^3)$ bits for an error probability of $2^{-k+\mathcal{O}(\log n)}$. For completeness, their construction π_{TVSS} is presented in Section 5.4.

In [CDD+99], Cramer et al. prove their construction is information-theoretic secure against adaptive corruptions under the classical definition of security (Definition 2.2.1). The next proposition shows that their protocol can be proven a secure Terminating VSS in the UC hybrid model we consider here if the model includes the synchronous communication (with guaranteed delivery) functionality $\mathcal{F}_{SYN}$ proposed in [Can05]. The proof is postponed until Section 5.4.

[6] If the model is extended to allow digital signatures, the round complexity can be reduced by half, but the resulting scheme is secure only against computationally-bounded adversaries.

Proposition 5.3.2 Protocol Π_{TVSS} UC-securely realizes $\mathcal{F}_{TVSS}$ in the ($\mathcal{F}_{BC}, \mathcal{F}_{SMT}$, $\mathcal{F}_{SYN}$)-hybrid model for $n > 2t$. ∎

Other Alternatives: Other VSS schemes than can potentially be used to achieve UC-TVSS are the ones presented by Goldreich et al. [BGW88], and Feldman and Micali [FM97]. They achieve resilience $n > 3t$ and constant-round complexity but they are less communication efficient than the protocol by Cramer et al.Also, the VSS scheme proposed by Rabin and Ben-Or [RBO89] with resilience $n > 2t$ (which is slightly less efficient than the one in [CDD+99]) can also be used. In the context of the computational variant of the UC model (where security holds against computationally bounded adversaries), Abe and Fehr [AF04] introduce the notion of *committed VSS*, in which the VSS protocol generates a commitment to the secret during the sharing phase. They present a very round and communication efficient protocol based on the Feldman's scheme [Fel87], and another (less efficient) based on the Pedersen's scheme [Ped91]. Unfortunately, the protocol based on Feldman's VSS does not suffice for our application as it leaks a Feldman commitment on the secret. The protocol based on Pedersen's VSS, however, can be proven UC-TVSS, but since it needs to generate a trapdoor-commitment key in a distributed way (see [AF04]), the resulting scheme is less efficient than the construction used above.

5.4 Adaptively secure VSS of Cramer et al.

In [CDD+99], Cramer et al. presented a very efficient adaptively secure VSS protocol in the secure-channel model. In this section, we describe the protocol and we prove it is also UC-TVSS. This section is adapted from Cramer et al. [CDD+99]. Let k be a security parameter, and $K = GF(q)$ a finite field such that $q > \max(n, 2^k)$.

5.4.1 Information Checking Protocol

An information checking protocol is the information-theoretic analogous of a "digital signature". Informally, an information checking protocol is a three party protocol that allows an intermediary I to obtain data from a dealer (or signer) D so later I can

convince a recipient D that the data actually comes from the dealer D. A more formal description follows, which is taken mostly verbatim from Cramer et al. [CDD+99].

Consider a protocol $\mathcal{IC} = (\mathcal{D}, \mathcal{A}, \mathcal{R})$ that consist of three phases (or subprotocols) executed between a dealer D, with input $s \in K$, an intermediary I, and a recipient R. The phases operate as follows.

- $\mathcal{D}$ is the *distribution* phase. This phase is initiated by the dealer D on input s. In this phase, D hands the secret s to the intermediary I and some auxiliary data to both I and the recipient R.
- $\mathcal{A}$ is the *authorization* phase. This phase is initiated by I and carried out by D, I, and R using as inputs the outputs obtained in the previous phase. In this phase, I ensures that in the protocol $\mathcal{R}$, the recipient R (if honest) will accept s, the secret held by I.
- $\mathcal{R}$ is the *reveal* phase. This phase is initiated by I and carried out by I and R. In this phase, R receives a value s' from I, along with some auxiliary data, and either accepts s' or rejects it.

We say protocol $\mathcal{IC}$ is a *secure information checking* protocol if satisfies the following properties

Correctness: If D, I, and R are honest, and D has a secret s then R will accept s in phase $\mathcal{R}$.

Binding-D: If I, and R are honest then after the phases $\mathcal{D}$ and $\mathcal{A}$, I knows a value v such that R will accept v in the phase $\mathcal{R}$ (except with probability 2^{-k}).

Binding-I: If D, and R are honest then in phase $\mathcal{R}$, with probability at least $1 - 2^{-k}$, party R will reject every value s' different than s.

Secrecy: The information that D hands R in phase $\mathcal{D}$ is distributed independently of the secret s. (Consequently, if D and I are honest, and I has not executed $\mathcal{R}$, then R has no information on the secret s.)

THE CDDHR INFORMATION CHECKING PROTOCOL: Cramer et al. [CDD+99] (building on [RBO89]) proposed a very efficient information checking protocol which assumes synchronous communication and secure channels. Before describing it, we need a definition. A vector $\mathbf{e} = (a, b, c) \in K^3$ is 1_α-consistent, denoted $\mathbf{e} \in R_{1_\alpha}$, if there exists a degree 1 polynomial $w(x)$ of degree at most t such that $w(0) = a$, $w(1) = b$, and $w(\alpha) = c$.

The proposed information checking protocol $\mathcal{IC} = (\texttt{Distr}, \texttt{AuthVal}, \texttt{RevealVal})$ is shown in Figures 5.2, 5.3 and 5.4. We adopt the convention that, at the end of each protocol step, the parties running the protocol (i.e. D,I, and R) invoke $\mathcal{F}_{SYN}$.[7]

Sub-Protocol $\texttt{Distr}(D, I, s)$

Public Input: field K description, session identifier sid^*

Private Input: D holds $s \in K$. I and R have no inputs.

(1) The dealer D chooses a random value $\alpha \in K \setminus \{0, 1\}$ and additional random values $r, r', s' \stackrel{R}{\leftarrow} K$ and computes $y \leftarrow r + s$, $y' \leftarrow r' + s'$, $z \leftarrow r\alpha + s$, and $z' \leftarrow r'\alpha + s'$. Notice that vectors (s, y, z) and (s', y', z') are 1_α-consistent. Then D sends (s, s', y, y') to the intermediary I, and (z, z', α) to the recipient R.

(2) The intermediary I's local output is (s, s', y, y'). The recipient R's local output is (z, z', α). The dealer's local output is $(s, s', y, y', z, z', \alpha)$.

Figure 5.2 Information Checking Protocol $\mathcal{IC}$ [CDD+99], Sub-Protocol `Distr`.

The following result is proven by Cramer et al. in [CDD+99].

Lemma 5.4.1 [CDD+99] The tuple $\mathcal{IC} = (\texttt{Distr}, \texttt{AuthVal}, \texttt{RevealVal})$ is a secure information checking scheme. ∎

IC-SIGNATURES: For notational simplicity, following [CDD+99] we cast the usage of an information checking protocol in the terminology of "digital signatures". Informally

[7] Each protocol step is done in a synchronous round, where the synchronicity is only for D,I, and R. Other parties may run asynchronously.

Sub-Protocol `AuthVal`(D, I, R)

Public Input: field K description, session identifier sid^*

Private Input: For each party, the local outputs obtained in `Distr`(D, I, s).

(1) Intermediary I chooses a random element $d \stackrel{R}{\leftarrow} K$, sets $(d, a, b) = (d, s'+ds, y'+dy)$ and broadcasts (d, a, b).

(2) In this step, there are two possible actions:

1. If dealer D sees that the values broadcasted by I are incorrect (that is, $a \neq s' + ds$ or $b \neq y' + dy$), then D broadcasts $(\texttt{Corrupted}, I, s, y)$. In this case, R's local output is the vector $(s, y, z) \in R_{1_\alpha}$ where (s, y) are the ones broadcasted by D. The protocol ends here.
2. Recipient R broadcasts `AcceptAuth` if $(a, b, z'+dz) \notin R_{1_\alpha}$ and `RejectAuth` otherwise.

(Notice that the broadcast done by D and R in this step can be done in parallel.)

(3) Depending on whether R's actions, there are two cases:

1. If D sees that R has acted incorrectly, D broadcasts $(\texttt{Corrupted}, R, z, \alpha)$. Party I's local output is the vector $(s, y, z) \in R_{1_\alpha}$. The protocol ends up here.
2. If R broadcasted `RejectAuth`, and D did not claim him faulty, then D broadcast $(\texttt{CorrectValues}, s, y)$, and the broadcasted values will be using in the following. Recipient R adjust z so $(s, y, z) \in R_{1_\alpha}$.

(Notice that the broadcast done by D and R in this step can be done in parallel.)

(4) The intermediary I's local output is (s, y). The recipient R's local output is (z, α). The dealer's local output is empty.

Figure 5.3 Information Checking Protocol $\mathcal{IC}$ [CDD+99], Sub-Protocol `AuthVal`.

Sub-Protocol `RevealVal`(I, R)

Public Input: field K description, session identifier sid^*

Private Input: For each party, the local outputs obtained in `AuthVal`(D, I, R).

(1) The intermediary I broadcasts (s, y).

(2) The recipient R broadcasts `AcceptReveal` if $(s, y, z) \in R_{1_\alpha}$, and `RejectReveal` otherwise.

Figure 5.4 Information Checking Protocol $\mathcal{IC}$ [CDD+99], Sub-Protocol `RevealVal`

speaking, the execution of the first two phases of the information checking protocol between a dealer D and an intermediary I on some input m, is somewhat reminiscent of D giving I a digital signature of m, since I can now convince R that m was indeed a valid message. Clearly, information checking protocols do not achieve the properties of the digital signatures, but they suffice for the VSS application.

Giving a IC-signature: We say that the dealer D gives a IC-signature $\sigma_m(D, I)$ to intermediary I, if the dealer D and intermediary I run the two first phases of the $\mathcal{IC}$ protocol. This process can be seen as "signature generation and first-time verification", as D "produces a signature" on the message and simultaneously I verifies the validity of the "signature" for the first time .

More precisely, given a party D acting as dealer and a party I acting as intermediary, we define a protocol `ICSignVrfy` as shown in Figure 5.5. Notice that, after the execution of `ICSignVrfy`(D, I, s) only party I may be convinced of the validity of the signature. For notational simplicity, somewhat reminiscent of the generation of a digital signature, we write $\sigma_m(D, I) \leftarrow$ `ICSignVrfy`(D, I, m) to denote that $\sigma_m(D, I)$ is the concatenation of the local outputs of dealer D, intermediary I, and all parties $P_\ell \in \mathcal{P}$ after these parties execute protocol `ICSignVrfy`(D, I, m).

Revealing (and verifying) an IC-signature: We now define the protocol that a party I executes with all other parties to *reveal* a valid IC-signature. Given dealer D and intermediary I, and a local state $\sigma_m(D, P)$ (see above) we define protocol

`ICRevealVrfy` as shown in Figure 5.5.

Round Complexity of IC-Signatures: It is easy to see that, assuming broadcast channels and point-to-point secure channels, `ICSignVrfy` takes four rounds (three if no faults occur) and `ICRevealVrfy` takes two rounds.

5.4.2 The Verifiable Secret Sharing Protocol of Cramer et al.

The protocol uses the *information checking* subprotocol defined above. We need a definition first.

Definition 5.4.2 A vector $\mathbf{e} = (e_1, \ldots, e_n) \in K^n$ is t-consistent, denoted $\mathbf{e} \in R_t$, if there exists a polynomial $w(x)$ of degree at most t such that $w(i) = e_i$, for $1 \leq i \leq n$. ▮

The protocol proposed in [CDD+99] is described in Figures 5.6 and 5.7.

Conventions for the description of π_{TVSS}:

- At the end of steps S-1 through S-5 and R-1 we add the instruction: "Then, all parties call $\mathcal{F}_{SYN}$."
- When we write "P privately sends m" it means that P invokes $\mathcal{F}_{SMT}$ with input m, and when we write "P broadcast m" it means that P invokes $\mathcal{F}_{SB}$ with input m.
- In the protocol description, "All parties call $\mathcal{F}_{SYN}$" denotes the operation where each party P_i does the following: P_i first sends message $(\texttt{Send}, sid, \text{“}\perp\text{”})$ to $\mathcal{F}_{SYN}$, then sends $(\texttt{Receive}, sid)$ to $\mathcal{F}_{SYN}$, and finally receives $(\texttt{Received}, sid, r, L_P^{r-1})$ which P_i discards.
- Protocol π_{VSS} can also be shown adaptively secure in the $(\mathcal{F}_{SMT}, \mathcal{F}_{SYN})$-hybrid model since there are efficient implementations of $\mathcal{F}_{BC}$ in the $\mathcal{F}_{SMT}$-hybrid model. Yet again, we preserve it for clarity of exposition.
- When we write "Party D *gives an IC-signature* on $m_1, \ldots, m_n$ to party R." it means D runs n parallel executions of protocol $\texttt{ICSignVrfy}(D, R, m_1)$, ..., proto-

Protocols `ICSignVrfy` and `ICRevealVrfy`

Public Input: session identifier sid^*

Protocol $\texttt{ICSignVrfy}(D, I, s)$

Private Input: D holds $s \in K$. I and R hold no input.
Private Output: $\sigma_s(D, I)$

(1) D and I execute protocol $\texttt{Distr}(D, I, m)$, and

(2) For all parties $P_\ell \notin \{D, I\}$, parties D, I, and P_ℓ execute *in parallel* protocol $\texttt{AuthVal}(D, I, P_\ell)$ on the output resulting from the above execution of $\texttt{Distr}(D, I, m)$.

(3) Party D's local output is empty. Party I's local output is the vector $((s_\ell, y_\ell))_{\ell \in [n]}$ where (s_ℓ, y_ℓ) is I's local output in $\texttt{AuthVal}(D, I, P_\ell)$. For each $P_\ell \notin \{D, I\}$, party P_ℓ's local output is (z_ℓ, α_ℓ) which are P_ℓ's local output in $\texttt{AuthVal}(D, I, P_\ell)$.

$$\text{Set } \sigma_s(D, I) \leftarrow \langle (D, \emptyset), (I, (s_\ell, y_\ell))_{\ell \in [n]}), (P_\ell, (z_\ell, \alpha_\ell))_{\ell \in [n]} \rangle.$$

Protocol `ICRevealVrfy`

Private Input: $\sigma_m(D, I)$.

(1) Parse $\sigma_s(D, I)$ as $\langle (D, \emptyset), (I, (s_\ell, y_\ell))_{\ell \in [n]}), (P_\ell, (z_\ell, \alpha_\ell))_{\ell \in [n]} \rangle$. That is, the input for D is empty, for I is $(s_\ell, y_\ell))_{\ell \in [n]}$, and for $P_\ell \in \mathcal{P}$ is (z_ℓ, α_ℓ).

(2) For each party $P_\ell \notin \{D, I\}$, I executes in parallel protocol $\texttt{RevealVal}(I, P_\ell)$.

(3) If there are at least $t+1$ of these parallel runs in which P_ℓ broadcasts "accept", each party $P \in \mathcal{P}$ locally outputs $(\texttt{ValidICSignature}, D, I, m)$.

Figure 5.5 Protocols for "giving" and "revealing" IC-signatures.

Protocol π_{TVSS} – Sharing Phase

Private Inputs: $D = P_1$ holds $s \in X$ (the "secret")

Public Input: session identifier sid^*

Private Outputs for each party P_i: a value $s_i \in X$ (a "share")

S-1. The dealer D chooses a random polynomial $f(x, y)$ of degree at most t in each variable, such that $f(0, 0) = s$. Let $s_{i,j} = f(i, j)$, and $a_{k,i} = s_{k,i}$, $b_{i,k} = s_{i,k}$, for $k = 1, \ldots, n$. The dealer D *gives an IC-signature* on $a_{k,i}$ and $b_{i,k}$, for $k = 1, \ldots, n$, to each party P_i.

S-2. Each party P_i checks that the vectors $(a_{1,i}, \ldots, a_{n,i})$ and $(b_{i,1}, \ldots, b_{i,n})$ are t-consistent. If not P_i broadcasts $((a_{1,i}, \ldots, a_{n,i}), (b_{i,1}, \ldots, b_{i,n}))$. Then P_i *reveals the IC-signatures* for the broadcasted values.

If a party P_j *hears* a broadcast of inconsistent values with *correct dealer's signature* then D is disqualified and the sharing phase ends.

S-3. Each party P_i sends $a_{j,i}$ privately to P_j and *gives an IC-signature* on $a_{j,i}$ to P_j.

S-4. Each party P_i compares the value $a_{i,j}$ received from P_j to the value $b_{i,j}$ received from D. If there is any inconsistency (or no value is received, or the signature is invalid), P_i broadcasts $b_{i,j}$, and reveals the corresponding IC-signature by D.

S-5. Each party P_i checks if P_j broadcasted a value $b_{j,i}$ which is different than the value $a_{j,i}$ which P_i holds. If such a broadcast exists (and P_i hears a correct dealer's signature for the value), then P_i broadcasts $a_{j,i}$ and reveals its corresponding IC-signature.

S-6. If for an index pair (i, j) a party P_k hears two broadcasts with correct signatures from the dealer on different values, then D is disqualified and the sharing phase ends.

Figure 5.6 Adaptively secure VSS π_{TVSS} [CDD+99] in the $(\mathcal{F}_{BC}, \mathcal{F}_{SMT}, \mathcal{F}_{SYN})$-hybrid model, Sharing Phase.

Protocol π_{TVSS} – Reconstruction Phase

Private Inputs for party P_i:

(If $D = P_1$): secret $s \in X$ (the "secret")

(If any P_i): values $(a_{k,i})_{k\in[n]}$ and $(b_{i,k})_{k\in[n]}$ (the "shares")

Public Input: session identifier sid^*

Private Outputs for each party P_i: a value $y \in X$ (the "reconstructed secret")

R-1. Each party P_i broadcasts the values $b_{i,1}, \ldots, b_{i,n}$ and reveals the signature for value $b_{i,j}$ which P_i received from party P_j. (If the dealer was disqualified at any step before, the secret is set to a default value.)

R-2. Each party P_i checks whether party P_j's shares broadcasted in the previous step are t-consistent and all the signatures heard are correct. If not, then P_j is disqualified.

R-3. The values of all non-disqualified parties are taken and interpolated to compute the secret.

Figure 5.7 Adaptively secure VSS π_{TVSS} [CDD+99] in the $(\mathcal{F}_{BC}, \mathcal{F}_{SMT}, \mathcal{F}_{SYN})$-hybrid model, Reconstruction Phase.

col `ICSignVrfy`(D, R, m_n). Let $\sigma_m(D, R)$ denote the local output after executing `ICSignVrfy`(D, R, m_n).

- When we write "P *reveals the IC-signatures* for value m", it means party P executes protocol `ICRevealVrfy`$(\sigma_m(D, P))$.
- When we write "P *hears* a value m with *correct signature by* D", it means P's local output in the last execution of `ICRevealVrfy`$(\cdot)$ was `ValidICSignature` (see protocol `ICRevealVrfy` ini Figure 5.5).
- If multiples signatures are to be given, all protocols `ICSignVrfy` are run in parallel. The same for signature verification (ie. protocol `ICRevealVrfy`.)

We are now ready to analyze the security of π_{TVSS}, proving Proposition 5.3.2.

Proof of Proposition 5.3.2: The proof is a simple extension of the proof presented in [CDD+99]. To prove UC security, we present an ideal adversary S that given any adaptive adversary A, is able to perfectly simulate A's view without the secret. Indeed, it suffices that S does the following. First, S picks t vectors $\mathbf{e}_1, \ldots, \mathbf{e}_t$ uniformly at random in K^{n+1} subject to that each $\mathbf{e}_i = (e_{i,k})_{0 \leq k \leq n}$ is t-consistent. These t vectors represent t arbitrary columns of the $(n+1) \times (n+1)$ matrix where the i, j-th entry is the value $f(i, j)$, for a (still arbitrary) bivariate polynomial f. Without lost of generality, assume $\mathbf{e}_1, \ldots, \mathbf{e}_t$ are the columns numbered 1 through t of this matrix. Then, S picks t random values $d_{0,1}, \ldots, d_{0,t} \in K$ and interpolate a vector $\mathbf{d}_k$ using Lagrange interpolation from values $d_{0,k}, e_{1,k}, \ldots, e_{t,k}$. Since each new vector $\mathbf{d}_k \in K^{n+1}$ is t-consistent and matches in t coordinates with vector $\mathbf{e}_i$, for $i = 1, \ldots, t$, this process generates t complete rows of the matrix (wlog, rows numbered 1 through t). Then, S hands vector pairs $(\mathbf{e}_1, \mathbf{d}_1), \ldots, (\mathbf{e}_{t'}, \mathbf{d}_{t'})$ to the $t' < t$ statically corrupted parties (those corrupted before the protocol execution starts) as the values $a_{k,i}, b_{i,k}$ of step (S-1). Similarly, S hands out $(\mathbf{e}_{t'+1}, \mathbf{d}_{t'+1}), \ldots, (\mathbf{e}_t, \mathbf{d}_t)$ to the parties subsequently corrupted by A. Meanwhile, the ideal adversary can simulate all honest (uncorrupted) parties without problems, as each value sent by a honest party P_i to a corrupted party P_j (namely $a_{i,j}$) is already part of the corrupted party P_j's t-consistent vectors distributed initially. Moreover, as long as the party acting as the intermediary for the IC-signatures is honest, S can "forge"

a correct signature by running the information checking protocol (`Distr` and `AuthVal`) using an arbitrary value as the secret. The secrecy property of the IC-signatures (more precisely, of the information checking protocol) guarantees that no corrupted party can distinguish a correct IC-signature from a forged one. Also, honest parties never complain about any other honest party so the simulation can proceed using only values in the $2t$ vectors $\mathbf{e}_1, \ldots, \mathbf{e}_t, \mathbf{d}_1, \ldots, \mathbf{d}_t$ until step (S-6). At that moment, S sends a message `DoEndSharing`, and waits until it receives message `Opened` containing the secret s. At this point, S waits until A sends message `Advance-Round`, and then simulates the execution as if the honest parties had sent empty inputs to the functionality $\mathcal{F}_{SYN}$, which causes step (S-6) to conclude. The rest of the simulation – the reconstruction phase – can be simulating by interpolating the bivariate polynomial $f(x, y)$ from the $2nt - t^2 + 1 \geq (t+1)^2$ precalculated values of the matrix (including the one in position $0, 0$ which is $f(0, 0) = s$). From the interpolated polynomial $f(x, y)$, S computes the shares (and vectors $(a_{k,i})_{k\in[n]}, (b_{i,k})_{k\in[n]}$) corresponding to each honest party P_i and honestly simulates reconstruction. The correctness of the simulation follows from the correctness and termination properties proved in [CDD+99]. ∎

5.5 UC Simultaneous Broadcast (UC-SB)

In this section, we generalize the simulation-based definition of Simultaneous Broadcast put forward by Chor et al. [CGMA85] to the UC framework. We achieve this by providing a *simultaneous broadcast* functionality $\mathcal{F}_{SB}$, which is a variant of the synchronous functionality [Can05] that provide "fairness", in the sense that the adversary is not allowed rushing. The functionality $\mathcal{F}_{SB}$ is shown in Figure 5.8. We say a protocol π achieves UC-SB if π UC-securely implements functionality $\mathcal{F}_{SB}$.

Intuitively, the definition of $\mathcal{F}_{SB}$ guarantees independence as the adversary cannot access any honest party's input until the broadcast is authorized to proceed, when it is "too late". Notice also that the functionality guarantees output delivery. In some applications, it may be useful to relax this condition.

UC-SB AND PREVIOUS SIMULTANEOUS BROADCAST DEFINITIONS: It is not hard to

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

$\mathcal{F}_{SB}$ expects its SID to be of the form $sid = (sid', \mathcal{P})$, where $\mathcal{P}$ is a list of parties among which broadcast is to be performed. It proceeds as follows.

(1) Upon receiving input $(\texttt{Broadcast}, sid, m)$ from party $P \in \mathcal{P}$, record (P, m) and output (sid, P) to the adversary. (If P later becomes corrupted then the record (P, m) is deleted.)

(2) Upon receiving message $(\texttt{Proceed}, sid, N)$ from the adversary, do: If there exist uncorrupted parties $P \in \mathcal{P}$ for which no record (P, m) exists then ignore the message. Else:

1. Interpret N as the list of messages sent by corrupted parties. That is, $N = \{(S_i, m_i)\}$ where each $S_i \in \mathcal{P}$ is corrupted, and m_i is a message.
2. Prepare a vector $\mathbf{m} = (m_i)_{i \in \mathcal{P}}$ of messages sent by all parties in $\mathcal{P}$, both corrupted and honest.

(3) Send $(\texttt{Broadcast}, sid, \mathbf{m})$ to the adversary.

(4) Upon receiving input $(\texttt{Receive}, sid)$ from a party $P \in \mathcal{P}$, output $(\texttt{Received}, sid, \mathbf{m})$ to P.

Figure 5.8 The simultaneous broadcast functionality, $\mathcal{F}_{SB}$.

see that UC-SB implies the (stand-alone) simulation-based definition of simultaneous broadcast in [CGMA85]. This is immediate since UC security implies stand-alone security [Can01b]. Then, by the results of the previous chapter, it holds that UC-SB implies all the other notions of Simultaneous Broadcast [CR87, Gen00].

5.5.1 A Generic Construction of UC-SB from UC-TVSS

In this section, we present our main construction. We show how to implement simultaneous broadcast (UC-SB) using Terminating VSS (UC-TVSS). The construction is simple: each party first runs the share phase of the TVSS in parallel; once all sharings have concluded (terminated), each party starts the reconstruction phase, gather all other parties' secrets and output the vector of values. (see Figure 5.9). Moreover, the construction works for any t; the final condition of honest majority comes from instantiating $\mathcal{F}_{TVSS}$ with π_{TVSS} (Proposition 5.3.2). Protocol π_{SB} is described in Figure 5.9.

Theorem 5.5.1 Protocol π_{SB} UC-securely realizes $\mathcal{F}_{SB}$ in the $\mathcal{F}_{TVSS}$-hybrid model for any t. ∎

Proof: Let A be a real-life adversary for π_{SB}. Note that A expects to interact with n parties running π_{SB} with access to n copies of functionality $\mathcal{F}_{TVSS}$. Given A, the ideal adversary S simulates the execution of protocol π_{SB} for adversary A by simulating the parties and functionalities as follows. Let $P_1, \ldots, P_n$ denote the simulated parties, $\tilde{P}_1, \ldots, \tilde{P}_n$ the ideal-world parties. Let sid^* be the session identifier under which each (simulated) party is first invoked (by the environment $\mathcal{Z}$), and $\mathcal{F}^1_{TVSS}, \ldots, \mathcal{F}^n_{TVSS}$ be the (simulated) n copies of functionality $\mathcal{F}_{TVSS}$, where $\mathcal{F}^k_{TVSS}$ denotes the functionality invoked by P_k with session identifier $sid_k = (sid^*, P_k, P_{[n]})$.[8] Adversary S maintains a set N with the corrupted parties and their inputs, initially $N \leftarrow \emptyset$, and proceeds as follows. If A corrupts any party P_i before the party has submitted a `Share` message to $\mathcal{F}^i_{TVSS}$ then S corrupts ideal-world party $\tilde{P}_i$, obtains its input x_i, and pass it to A. If A instructs corrupted party P_i to submit $(\texttt{Share}, x'_i)$ to $\mathcal{F}^i_{TVSS}$, then S simulates the

[8] Notice that, such functionality may also be invoked (and instantiated) by some other party P_j on message `EndSharing` if P_k is not activated by $\mathcal{Z}$.

Protocol π_{SB}

Private Inputs: Each P_i holds $x_i \in X$

Public Input: session identifier sid^*

Private Outputs: a vector $\mathbf{y}_i \in (X \cup \{\perp\})^n$ for each P_i

Each party $P_i \in \mathcal{P}$ runs sequentially the following steps:

(1) For each $j \in P_{[n]}$, set $sid_j \leftarrow (sid^*, P_j, P_{[n]})$.

(2) Send $(\texttt{Spool}, sid_i, x_i)$ to $\mathcal{F}_{TVSS}$.

(3) Send $(\texttt{Share}, sid_i, x_i)$ to $\mathcal{F}_{TVSS}$.

(4) For each $j \in P_{[n]}$, send $(\texttt{EndSharing}, sid_j)$ to $\mathcal{F}_{TVSS}$.

(5) Upon receiving $(\texttt{Shared}, sid_j)$ from $\mathcal{F}_{TVSS}$, record $(\texttt{Shared}, sid_j)$. Repeat this step until there is a record $(\texttt{Shared}, sid_j)$ for each $j \in [n]$

(6) For each $j \in P_{[n]}$, send $(\texttt{Open}, sid_j)$ to $\mathcal{F}_{TVSS}$.

(7) Upon receiving $(\texttt{Opened}, sid_j, v_j)$ from $\mathcal{F}_{TVSS}$, record (sid_j, y_j). Once a record (sid_j, y_j) for each $j \in [n]$ exists, output vector $\mathbf{y}_i = (y_j)_{j \in [n]}$ and halt.

Figure 5.9 Simultaneous Broadcast protocol in the $\mathcal{F}_{TVSS}$-hybrid model.

operation, and adds $(\tilde{P}_i, x'_i)$ to N. For all uncorrupted parties P_k, S sets P_k's input to an arbitrary value (eg. $x'_k \leftarrow \perp$) and simulates P_k's interaction with $\mathcal{F}^k_{TVSS}$ by simulating both, party and functionality. Notice that S can do such simulation without the real P_k's input because adversary A's view of the interaction between P_k and $\mathcal{F}^k_{TVSS}$ during the *share* phase of TVSS (steps (1)-(6) of Figure 5.1) is independent of P_k's input. Indeed, consider the event E_k defined as "$\mathcal{F}^k_{TVSS}$ has at least one record $(\texttt{EndSharing}, P)$ *and then* it receives a message `DoEndSharing` from A". As long as P_k is corrupted anytime before E_k is true, S can proceed as before, that is, S obtains x_k from corrupting $\tilde{P}_k$ and pass it to A. Notice, however, that adversary A must corrupt a party P_i before P_i sends out message `Share` to $\mathcal{F}^i_{TVSS}$ if A wants to change the value submitted by P_i.

At some point in the simulation, A may send a `DoEndSharing` message to some TVSS functionality. Then, S partitions the simulated parties in four sets. These sets are dynamic in the sense that S may *move* parties from one set to another depending on the subsequent instructions of A. The corrupted parties are partitioned into B_{Sh} and its complement, where B_{Sh} is the set of parties which have submitted a message `Share` to $\mathcal{F}_{TVSS}$. (Notice that for all $P_i \in B_{Sh}$, N contains an entry $(\tilde{P}_i, x'_i)$.) Similarly, any honest party P_i is either in G_{Sh} or its complement, where G_{Sh} is the set of parties that have submitted a message `Shared` to its $\mathcal{F}^i_{TVSS}$. Notice that if $P_i \in G_{Sh}$, then P_i has sent or is about to send a `EndSharing` message. Let $B_{\texttt{end}}$ (resp. $G_{\texttt{end}}$) be the set of corrupted (resp. uncorrupted) parties whose corresponding TVSS functionalities have at least one record of the form $(\texttt{EndSharing}, P_j)$. Assume A sends a message `DoEndSharing` to functionality $\mathcal{F}^k_{TVSS}$. Then,

(1) If $P_k \notin B_{\texttt{end}} \cup G_{\texttt{end}}$, that is, $\mathcal{F}^k_{TVSS}$ has no `EndSharing` records, then S does nothing (since those messages would be ignored by the TVSS functionality).

(2) If $P_k \in G_{\texttt{end}}$ but $P_k \notin G_{Sh}$, that is, $\mathcal{F}^k_{TVSS}$ has one or more `EndSharing` records but P_k has yet to submit a `Share` request to $\mathcal{F}^k_{TVSS}$, then S does nothing (since those messages would be ignored by the TVSS functionality).

(3) If $P_k \in B_{\texttt{end}} \cap B_{Sh}$ or if $P_k \in G_{\texttt{end}} \cap G_{Sh}$, then S simulates $\mathcal{F}^k_{TVSS}$'s execution by having the functionality send messages $(\texttt{Shared}, sid_k)$ to all parties P_i and the adversary A. If $P_k \in B_{\texttt{end}} \cap \overline{B_{Sh}}$, then S does the same but also adds $(P_k, \perp)$ into

set N.

(4) If A instructs a corrupted party P_i to send a message `Open` to some $\mathcal{F}_{TVSS}^k$, then S honestly simulates the functionality.

We also let $J \subseteq (G_{\texttt{end}} \cap G_{Sh}) \cup B_{\texttt{end}}$ be the set of parties to whose functionality A has sent a message `DoEndSharing`. S continues the simulation following the above rules (possibly moving parties into G_{Sh}, B_{Sh}, $G_{\texttt{end}}$, $B_{\texttt{end}}$, and J as new messages are delivered by A) until $J = [n]$. Assume this happens when A sends a message `DoEndSharing` to $\mathcal{F}_{TVSS}^k$. *Before* applying rule 3 from above, S sends $(\texttt{Proceed}, sid^*, N)$ to ideal functionality $\mathcal{F}_{SB}$, and obtains $(\texttt{Broadcast}, sid^*, \mathbf{m})$. S uses $\mathbf{m}$ to set the secret in each simulated (uncorrupted) $\mathcal{F}_{TVSS}^i$ to $s_i = m_i$, where $\mathbf{m} = (m_1, \ldots, m_n)$. Only then S applies rule 3 from above for party P_k. From then on, S honestly simulates the execution of π_{SB} for A.

We claim that the simulation is perfect. Indeed, observe that adversary A's view before set J becomes equal to $[n]$ is independent of the input of the simulated parties, as it consists of the corrupted parties' inputs, and messages $(\texttt{SpoolRcvd}, sid_i, P_i)$, $(\texttt{ShareRcvd}, sid_i, P_i)$, $(\texttt{EndSharingRcvd}, sid_i, P_i)$, and $(\texttt{Shared}, sid_j, P_i)$ for one or more party $P_i \in (G_{\texttt{end}} \cap G_{Sh}) \cup B_{\texttt{end}}$. The crucial observation is that no uncorrupted party P_k issues an `Open` message unless P_k has received `Shared` messages for all parties. This only happens if `DoEndSharing` messages have been received by each functionality $\mathcal{F}_{TVSS}^j$, $P_j \in J$, which only happens *after* J is set to $[n]$. At that point, the ideal adversary S has obtained the inputs for all parties, so the adversary's view from then on is identical to the real-world experiment. Notice also that once adversary A sends `DoEndSharing` messages to each functionality $\mathcal{F}_{TVSS}^j$, $P_j \in J = [n]$, A cannot issue a `Share` message for any (corrupted) party P_i. This is because P_i must also be in $J \subseteq (G_{\texttt{end}} \cap G_{Sh}) \cup B_{\texttt{end}}$ which implies P_i is either in $G_{\texttt{end}} \cup B_{\texttt{end}}$, and therefore functionality $\mathcal{F}_{TVSS}^i$ has successfully executed step (6) where $(\texttt{Shared}, sid_i)$ was sent out to all parties; after this step, no new `Share` or`DoEndSharing`input is accepted by $\mathcal{F}_{TVSS}^i$. This concludes the proof. ▮

On the Synchronicity of Simultaneous Broadcast and TVSS: We conclude this section showing that requiring Simultaneous Broadcast is essentially as strong as

requiring synchronous communication, namely $\mathcal{F}_{SYN}$ [Can05]. One direction is provided by the reduction to UC-TVSS described above, which says that any solution for UC-TVSS can be used to achieve UC-SB. Notice also that $\mathcal{F}_{SYN}$ implies UC-TVSS by Proposition 5.3.2. The other direction follows from the fact UC-SB can be used to implement $\mathcal{F}_{SYN}$ as follows: first parties (non-simultaneously) broadcast their values, and then use simultaneous broadcast to transmit the same values (i.e. those broadcasted non-simultaneously before). Thus, the following claim holds.

Claim 5.5.2 Let π be a protocol that UC-securely realizes $\mathcal{F}_{SB}$ in the $\mathcal{F}_{SMT}$-hybrid model. Then, there exists a protocol that UC-securely realizes $\mathcal{F}_{SYN}$ in the $(\mathcal{F}_{SB}, \mathcal{F}_{SMT})$-hybrid model. ▮

5.6 Extensions

Removing secure channels: Our protocol for simultaneous broadcast is only secure in the secure channel model. To obtain a protocol secure in the public channel model (i.e. authenticated channels), we can use known techniques, like those proposed by Lysyanskaya [Lys00] which require secure erasures. Another alternative is to use non-committing encryption [CFGN96], in particular the efficient scheme of Damgård and Nielsen [DN00]. For the case of static corruption is much simpler, as encrypting the messages with a semantic secure encryption scheme suffices.

Asynchronous Simultaneous Broadcast (UC-ASB): It is well known that in a synchronous network, no functionality that depends on all the inputs can be computed [PSL80, CM89]. This is because it is impossible to distinguish between failed processes (that is, those instructed to not send messages) and very slow processes. Therefore, no process can afford to wait for messages coming from more than $n - t$ distinct other processes. In this section, we adapt the functionality of Simultaneous Broadcast to comply with this restriction, at the cost of weakening the guarantee that all players can participate in the broadcast (which is unavoidable). We remark that, nonetheless, the modified functionality still preserves the intuitive notion of *independence*, as long as parties that do not participate in the broadcast are not allowed to contribute later with

their inputs. The corresponding functionality $\mathcal{F}_{ASB}$ is shown in Figure 5.10. We say a protocol π achieves UC-ASB if π UC-securely realizes functionality $\mathcal{F}_{ASB}$.

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

$\mathcal{F}_{SB}$ expects its SID to be of the form $sid = (sid', \mathcal{P}, t)$, where $\mathcal{P}$ is a list of parties among which broadcast may potentially be performed, and $t < n$ is an integer, where $n \stackrel{\text{def}}{=} |\mathcal{P}|$. It proceeds as follows.

(1) Upon receiving input $(\texttt{Broadcast}, sid, m)$ from party $P \in \mathcal{P}$, record (P, m) and output (sid, P) to the adversary. (If P later becomes corrupted then the record (P, m) is deleted.)

(2) Upon receiving a message $(\texttt{Proceed}, sid, N, W)$ from the adversary, do: If W is a subset of parties in $\mathcal{P}$ of size at least $n - t$, and there exist uncorrupted parties $P \in W$ for which no record (P, m) exists then ignore the message. Else:

1. Interpret N as the list of messages sent by corrupted parties. That is, $N = \{(S_i, m_i)\}$ where $S_i \in W$ and S_i is corrupted, and m_i is a message.
2. Prepare a vector $\mathbf{m} = (m_i)_{i \in W}$ of messages sent by all parties in W, both corrupted and honest.

(3) Send $(\texttt{Broadcast}, sid, \mathbf{m})$ to the adversary.

(4) Upon receiving input $(\texttt{Receive}, sid)$ from a party $P \in \mathcal{P}$, send $(\texttt{Received}, sid, \mathbf{m})$ as delayed output to P.

Figure 5.10 The asynchronous simultaneous broadcast functionality, $\mathcal{F}_{ASB}$.

We claim (without proof) that there exists a simple construction that achieves UC-ASB for the case $n > 3t$. It suffices to run first the initial phase of the secure multiparty computation of Ben-Or et al. [BOKR94]. Spelled out, first, parties run n parallel copies of the *ultimate secret sharing* protocol; then the protocol for *agreement on a common subset* is run. (Both protocols are described in [BOKR94].) In this way, all parties agree on the set W of parties that have properly shared their input.

The reconstruction protocol is executed next, where the secrets of all parties in W is reconstructed. For computationally bounded adversaries, a similar approach can be obtained using the initialization phase of the computationally efficient construction of [HNP05]. It is an open problem whether more communication efficient solutions exist.

REFERENCE: Most material in this chapter is a reprint from the material appearing in "Universally Composable Simultaneous Broadcast", A. Hevia, in the proceedings of the 5th Security and Cryptography for Networks (SCN 2006), LNCS, Springer-Verlag, 2006.

Appendix A

Appendix

A.1 Alternative Characterization of Simultaneous Broadcast Notions

A.1.1 Sb-Independence

We prove here that Sb-Independence is equivalent to (All, Sb)-Independence.

Proposition A.1.1 A protocol Π achieves Sb-Independence if and only if Π achieves (All, Sb)-Independence. ∎

Proof: One direction is trivial since Singleton $\subset$ All. We prove the other direction, namely that (Singleton, Sb)-Independence implies (All, Sb)-Independence. Assume Π is Sb-Independent. Then, for every polynomial-time adversary A attacking Π there exists a simulator S for $\mathsf{Ideal}(f_{SB})$. Then, by conditioning on the success probability of the distinguisher T on each particular value of the input distribution we get

$$\begin{aligned}
&\Big|\Pr\Big[\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)} \ : \ T(1^k, z, \mathbf{x}, \textsc{Exec}_A^{\Pi}(k, z, \mathbf{x})) = 1\Big] \\
&\qquad - \Pr\Big[\mathbf{x} \stackrel{R}{\leftarrow} \mathcal{D}^{(k)} \ : \ T(1^k, z, \mathbf{x}, \textsc{Exec}_S^{\mathsf{Ideal}(f_{SB})}(k, z, \mathbf{x})) = 1\Big]\Big| \\
&= \sum_{\mathbf{y}\in\{0,1\}^n} \Big(\Pr\Big[T(1^k, z, \mathbf{x}, \textsc{Exec}_A^{\Pi}(k, z, \mathbf{y})) = 1\Big]
\end{aligned}$$

$$- \Pr\left[T(1^k, z, \mathbf{x}, \text{EXEC}_S^{\mathsf{Ideal}(f_{SB})}(k, z, \mathbf{y})) = 1\right]\Big) \cdot \Pr\left[\mathcal{D}^{(k)} = \mathbf{y}\right]$$

$$\leq \sum_{\mathbf{y}\in\{0,1\}^n} \left(k^{-c}\right) \cdot \Pr\left[\mathcal{D}^{(k)} = \mathbf{y}\right] \;=\; k^{-c}\,.$$

for some constant $c > 0$ and infinitely many values of k. Notice that the last inequality follows from the Sb-Independence of Π. This proves the result. ▮

A.1.2 CR-Independence

In this section, we present the definition of independence of [CR87], slightly generalized for arbitrary input distributions.

Definition A.1.2 [CR87] Let $\mathcal{D}$ be an input distribution over $\{0,1\}^n$. A protocol Π achieves independence under input distribution $\mathcal{D}$ if, for all adversary A, all honest party P_i, all "good" polynomial-time predicate R, all constant $c > 0$, and all sufficiently large k,

$$\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] - \Pr\left[\,W_i = 0 \mid\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \;<\; k^{-c} \tag{A.1}$$

where $\mathbf{W} \leftarrow \text{ANNOUNCED}_A^{\Pi}(\mathcal{D})$. Predicate R is "good" if it occurs with non-negligible probability under adversary A and distribution $\mathcal{D}$. Formally, for $i \in [n]$ the polynomial-time computable predicate $R(W_1, \ldots,\; W_{i-1}, W_{i+1}, \ldots, W_n)$ is said to be *good* with respect to adversary A and distribution $\mathcal{D}$ if whenever party P_i is honest the event $R(\mathbf{W}_{\overline{\{i\}}}) = 1$ happens with non-negligible probability. That is,

$$\Pr\left[\,\mathbf{W} \leftarrow \text{ANNOUNCED}_A^{\Pi}(\mathcal{D}^{(k)}) \;:\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]$$

is non-negligible in the security parameter k. ▮

We now show that Definition 4.3.3 and Definition A.1.2. are in fact equivalent.

Proposition A.1.3 A protocol Π achieves CR-Independence if and only if Π achieves independence under Definition A.1.2. ▮

Proof: We first prove that Definition 4.3.3 implies Definition A.1.2. Let Π be a multi-sender broadcast that achieves CR-Independence. It follows that, in particular, for all

"good" predicates R

$$\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,W_i = 0 \;\wedge\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right|$$

is negligible. Using this and that R is good, we get that the quantity

$$\begin{aligned}
&\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \mid\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \\
&= \quad \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]^{-1} \cdot \\
&\qquad \left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \;\wedge\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right|
\end{aligned}$$

is negligible too.

We now prove that Definition A.1.2 implies Definition 4.3.3. Indeed, assume Π be a multisender broadcast that achieves independence according to Definition A.1.2. We analyze four cases depending on the probability p that the event $R(\mathbf{W}_{\overline{\{i\}}}) = 1$ (when $\mathbf{W} \leftarrow \textsc{Announced}_A^{\Pi}(\mathcal{D}^{(k)})$) occurs.

Case (a): p is negligible but non-zero, Then,

$$\begin{aligned}
&\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \;\wedge\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \\
&= \quad \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] \cdot \left|\Pr\left[\,\mathbf{W}_i = 0\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \mid\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \qquad \text{(A.2)}
\end{aligned}$$

is negligible, since $p = \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]$ is so.

Case (b): p is non-negligible. Since Π satisfies Definition A.1.2, the rightmost factor of Equation A.2 is negligible too.

Case (c): $p = 0$, that is, R does never happen. Then, $\Pr\left[\,\mathbf{W}_i = 0 \;\wedge\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] = 0$, and

$$\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \;\wedge\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \;=\; 0\,.$$

Case (d): p is neither negligible nor non-negligible. We prove the contrapositive. If Definition 4.3.3 does not hold for such R, there exist a constant $c > 0$ such that for infinitely many values of k

$$\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \;\wedge\; R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \;\geq\; k^{-c} \qquad \text{(A.3)}$$

Let S be the set of all values of k for which Equation (A.3) holds. Now, consider the following relation R' which equals R when $k \in S$ and equals one (ie. it is true) otherwise. Then, clearly R' is non-negligible and

$$\begin{aligned}
&\left|\Pr\left[\,\mathbf{W}_i = 0\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \mid\ R'(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \\
&= \Pr\left[\,R'(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]^{-1} \cdot \\
&\qquad \left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R'(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \ \wedge\ R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \\
&\geq \left|\Pr\left[\,\mathbf{W}_i = 0\,\right] \cdot \Pr\left[\,R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right] - \Pr\left[\,\mathbf{W}_i = 0 \ \wedge\ R(\mathbf{W}_{\overline{\{i\}}}) = 1\,\right]\right| \\
&\geq k^{-c}
\end{aligned}$$

for all (infinitely many) $k \in S$. The result then holds. ∎

A.1.3 G-Independence

In this section, we present two equivalent notions of independence, and then show they imply G-Independence. Our first definition is expressed in terms of distributions ensembles.

Definition A.1.4 (G*-Independence) Protocol Π achieves G*-independence if for all adversaries A corrupting parties in $B \subset [n]$ (where $|B| = t < n$), for each corrupted party P_i, the ensembles (indexed by $k \in \mathbf{N}$, $\mathbf{x} \in \{0,1\}^n$, and $z \in \{0,1\}^*$)

$$\begin{aligned}
E &\stackrel{\text{def}}{=} \left\{\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{x}) \ :\ \mathbf{W_i}\right\} \\
E_0 &\stackrel{\text{def}}{=} \left\{\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{x}_B \sqcup \langle \mathbf{0} \rangle_{\overline{B}}) \ :\ \mathbf{W_i}\right\}
\end{aligned}$$

are statistically close (in the security parameter k). ∎

Our second definition, although more technical, is useful when proving implications or separations between G and other notions.

Definition A.1.5 (G**-Independence) Protocol Π achieves G**-independence if for all adversaries A corrupting parties in $B \subset [n]$ (where $|B| = t < n$), for each corrupted

party P_i, for all vectors $\mathbf{r}, \mathbf{s} \in \{0,1\}^{\overline{B}}$, all vectors $\mathbf{w} \in \{0,1\}^B$, and all auxiliary input $z \in \{0,1\}^*$, the quantity

$$\Big|\Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{s}) : \mathbf{W_i} = 1\,\Big] - \Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{r}) : \mathbf{W_i} = 1\,\Big]\Big|$$

is negligible in the security parameter k. ▌

The two definitions are equivalent.

Proposition A.1.6 Let Π be a correct multisender broadcast. Then, Π achieves G^{**}-Independence if and only if Π achieves G^{*}-Independence. ▌

Proof:

$\mathsf{G}^{**} \Rightarrow \mathsf{G}^{*}$: Assume Π is not G^{*}-Independent. We want to prove Π is not G^{**}-Independent. Indeed, if Π is not G^{*}-Independent then ensembles E and E_0 must not be statistically close, and there exists $\mathbf{x} \in \{0,1\}^n$, $k \in \mathbf{N}$, and $z \in \{0,1\}^*$, for which there exists a constant $c > 0$ and infinitely many k such that (w.l.o.g.)

$$\Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{x}_B \sqcup \mathbf{x}_{\overline{B}}) : \mathbf{W_i} = 1\,\Big] - \Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{x}_B \sqcup \mathbf{0}_{\overline{B}}) : \mathbf{W_i} = 1\,\Big] > k^{-c}$$

The result follows immediately from taking $\mathbf{w} = \mathbf{x}_B$, $\mathbf{r} = \mathbf{x}_{\overline{B}}$ and $\mathbf{s} = \mathbf{0}_{\overline{B}}$.

$\mathsf{G}^{*} \Rightarrow \mathsf{G}^{**}$: Assume Π is not G^{**} independent. We want to prove Π is not G^{*}-Independent. Indeed, if Π is not G^{**}-Independent then there exists a vector $\mathbf{w} \in \{0,1\}^B$, distinct vectors $\mathbf{r}, \mathbf{s} \in \{0,1\}^{\overline{B}}$, an integer $k \in \mathbf{N}$, and a string $z \in \{0,1\}^*$, for which there exists a constant $c > 0$ and infinitely many k such that (w.l.o.g.)

$$\Big|\Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{s}) : \mathbf{W_i} = 1\,\Big] - \Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{w} \sqcup \mathbf{r}) : \mathbf{W_i} = 1\,\Big]\Big| > k^{-c} \qquad \text{(A.4)}$$

For simplicity, we define $D(\mathbf{a}) \stackrel{\text{def}}{=} \Pr\Big[\,\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{a}) : \mathbf{W_i} = 1\,\Big]$, for any vector $\mathbf{a} \in \{0,1\}^n$. Let $\mathbf{x} \stackrel{\text{def}}{=} \mathbf{w} \sqcup \mathbf{r}$ and $\mathbf{x}' \stackrel{\text{def}}{=} \mathbf{w} \sqcup \mathbf{s}$. Then, Equation (A.4) can be

rewritten as $|D(\mathbf{x}) - D(\mathbf{x}')| > k^{-c}$. In consequence, using that $\mathbf{x}_B = \mathbf{x}'_B = \mathbf{w}$ we have

$$\left|D(\mathbf{x}) - D(\mathbf{x}_B \sqcup \mathbf{0}_{\overline{B}})\right| + \left|D(\mathbf{x}') - D(\mathbf{x}'_B \sqcup \mathbf{0}_{\overline{B}})\right| \;\geq\; \left|D(\mathbf{x}) - D(\mathbf{x}')\right| \;>\; k^{-c}$$

which implies that either $|D(\mathbf{x}) - D(\mathbf{x}_B \sqcup \mathbf{0}_{\overline{B}})| > k^{-c}/2$ or $|D(\mathbf{x}') - D(\mathbf{x}'_B \sqcup \mathbf{0}_{\overline{B}})| > k^{-c}/2$, and the result follows. ▮

The following result shows that both G^* and G^{**}-Independence imply G-Independence for any distribution for which G can be achieved.

Proposition A.1.7 , If a protocol Π achieves G^{**}-Independence then Π achieves G-Independence for any distribution $\mathcal{D} \in \Psi_{L,n}$. ▮

Proof: Let $\mathcal{D}$ be an arbitrary distribution in $\Psi_{C,n}$ and $\mathbf{r}, \mathbf{s} \in \{0,1\}^{n-t}$ two strings such that the probability $\mathcal{D}_{\overline{B}}$ equals $\mathbf{r}$ or $\mathbf{s}$ is not null. Also, let A be an arbitrary polynomial-time adversary that corrupt players in B ($t = |B|$) and let $i \in B$. For fixed values of $k \in \mathbf{N}$ and $z \in \{0,1\}^*$, we denote by $\Pr_{\mathcal{D},A}[E]$ the probability of event E under the choice $\mathbf{x} \xleftarrow{R} \mathcal{D}^{(k)}$ and $\mathbf{W} \leftarrow \textsc{Announced}^{\Pi}_{A(k,z)}(\mathbf{x})$. Now, for simplicity, we define the quantities

$$\begin{aligned} P(\mathbf{a},\mathbf{b}) &\stackrel{\text{def}}{=} \Pr_{\mathcal{D},A}\left[\mathbf{W_i} = 1 \mid \mathbf{x}_B = \mathbf{a} \wedge \mathbf{x}_{\overline{B}} = \mathbf{b}\right] \\ Q(\mathbf{a},\mathbf{b}) &\stackrel{\text{def}}{=} \Pr\left[\mathbf{x} \xleftarrow{R} \mathcal{D}^{(k)} : \mathbf{x}_B = \mathbf{w} \mid \mathbf{x}_{\overline{B}} = \mathbf{b}\right]. \end{aligned}$$

First, notice that $\mathbf{W}_{\overline{B}} = \mathbf{x}_{\overline{B}}$, since all uncorrupted parties always output their inputs. Then

$$\begin{aligned} &\Pr_{\mathcal{D},A}\left[\mathbf{W_i} = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{r}\right] - \Pr_{\mathcal{D},A}\left[\mathbf{W_i} = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{s}\right] \\ &= \sum_{\mathbf{w} \in \{0,1\}^t} \left(P(\mathbf{w},\mathbf{r}) \cdot Q(\mathbf{w},\mathbf{r}) - P(\mathbf{w},\mathbf{s}) \cdot Q(\mathbf{w},\mathbf{s})\right) \end{aligned} \tag{A.5}$$

By G^{**}-Independence, for all $\mathbf{w}' \in \{0,1\}^t$, all $\mathbf{r}', \mathbf{s}' \in \{0,1\}^{n-t}$, $|P(\mathbf{w}',\mathbf{r}') - P(\mathbf{w}',\mathbf{s}')| < \epsilon(k)$ where $\epsilon(k)$ is some negligible function in k. Let $P(\mathbf{w},\mathbf{t}^*) \stackrel{\text{def}}{=} \max_{\mathbf{t}^*}\{P(\mathbf{w},\mathbf{t})\}$. Then, by definition it follows that $P(\mathbf{w},\mathbf{r}) \leq P(\mathbf{w}^*,\mathbf{t})$ and, by G^{**}-Independence, that $P(\mathbf{w},\mathbf{s}) < P(\mathbf{w},\mathbf{t}^*) + \epsilon(k)$. Then, plugging these in Equation (A.5) we have

$$\sum_{\mathbf{w} \in \{0,1\}^t} \left(P(\mathbf{w},\mathbf{r}) \cdot Q(\mathbf{w},\mathbf{r}) - P(\mathbf{w},\mathbf{s}) \cdot Q(\mathbf{w},\mathbf{s})\right)$$

$$\leq \sum_{\mathbf{w}\in\{0,1\}^t} (P(\mathbf{w},\mathbf{t}^*)\cdot(Q(\mathbf{w},\mathbf{r})-Q(\mathbf{w},\mathbf{r}))) + \epsilon(k) \tag{A.6}$$

Now, define $R(\mathbf{w},\mathbf{r},\mathbf{s}) \stackrel{\text{def}}{=} Q(\mathbf{w},\mathbf{r}) - Q(\mathbf{w},\mathbf{s})$. We claim that $|R(\mathbf{w},\mathbf{r},\mathbf{s})|$ is negligible in k. Indeed, since $\mathcal{D}\in\Psi_{L,n}$,

$$\begin{aligned}
|R(\mathbf{w},\mathbf{r},\mathbf{s})| &= |Q(\mathbf{w},\mathbf{r}) - Q(\mathbf{w},\mathbf{s})| \\
&= \left|Q(\mathbf{w},\mathbf{r}) - \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{w}\right] + \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{w}\right] - Q(\mathbf{w},\mathbf{s})\right| \\
&\leq \left|\Pr\left[\mathbf{x}\stackrel{R}{\leftarrow}\mathcal{D}^{(k)} : \mathbf{x}_B = \mathbf{w} \mid \mathbf{x}_{\overline{B}} = \mathbf{r}\right] - \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{w}\right]\right| \\
&\quad + \left|\Pr\left[\mathbf{x}\stackrel{R}{\leftarrow}\mathcal{D}^{(k)} : \mathbf{x}_B = \mathbf{w} \mid \mathbf{x}_{\overline{B}} = \mathbf{r}\right] - \Pr\left[\mathcal{D}_B^{(k)} = \mathbf{w}\right]\right| \\
&< 2\cdot\epsilon'(k)
\end{aligned} \tag{A.7}$$

for some negligible function $\epsilon'(k)$. Combining Equations $(A.5)$, $(A.6)$ and $(A.7)$ we obtain

$$\left|\Pr\left[\mathbf{W_i} = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{r}\right] - \Pr\left[\mathbf{W_i} = 1 \mid \mathbf{W}_{\overline{B}} = \mathbf{s}\right]\right| < 2\cdot\epsilon'(k) + \epsilon(k)\,.$$

This proves the result. ∎

Bibliography

[Abe98] M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 437–447. Springer–Verlag, 1998.

[AF04] M. Abe and S. Fehr. Adaptively secure feldman vss and applications to universally-composable threshold cryptography. In *Advances in Cryptology–CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 317–334. Springer-Verlag, 2004.

[BBDP01] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT' 2001*, Lecture Notes in Computer Science, pages 566–582. International Association for Cryptologic Research, Springer-Verlag, 2001.

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT ' 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer-Verlag, 2000.

[BCG93] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 52–61. ACM Press, 1993.

[BD03] A. Beimel and S. Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16, 2003.

[BdB89] J. Bos and B. den Boer. Detection of disrupters in the DC protocol. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology–EUROCRYPT ' 89*, volume 434 of *Lecture Notes in Computer Science*, pages 320–328. Springer-Verlag, 1990, 10–13 April 1989.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk,

editor, *Advances in Cryptology - CRYPTO '98, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[BF01] D. Boneh and M.K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BFTS04] R. Berman, A. Fiat, and A. Ta-Shma. Provable unlinkability against traffic analysis. In *International Conference on Financial Cryptography – FC'04*, volume 3110 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.

[BGW01] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *MOBICOM*, pages 180–189, 2001.

[BIK+03] M. Blaze, J. Ioannidis, A.D. Keromytis, T. Malkin, and A. Rubin. WAR: Wireless anonymous routing. In *Security Protocols Workshop*, volume 3364 of *Lecture Notes in Computer Science*, pages 218–232. Springer-Verlag, 2003.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979 National Computer Conference*, pages 313–317. AFIPS, 1979.

[BLR04] B. Barak, Y. Lindell, and T. Rabin. Protocol initialization for the framework of universal composability. Cryptology ePrint Archive, Report 2004/006, 2004. `http://eprint.iacr.org/`.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In Baruch Awerbuch, editor, *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing – STOC'90*, pages 503–513, Baltimore, MY, May 1990. ACM Press.

[BN00] M. Bellare and C. (Meaw) Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, 2000.

[BOEY03] M. Ben-Or and R. El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4), 2003.

[BOKR94] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, pages 183–192, 1994.

[BPS00] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[BR93] M. Bellare and P. Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 22–26 August 1993.

[BR04] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`.

[BTH06] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In Shai Halevi and Tal Rabin, editors, *3rd Theory of Cryptography Conference, TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer-Verlag, 2006.

[Can00a] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can00b] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01a] R. Canetti. A unified framework for analyzing security of protocols. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2001.

[Can01b] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, 2001.

[Can05] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Report 2000/067, Cryptology ePrint Archive, January 2005. Revised version of [Can01b]. Updated Jan. 28, 2005.

[CCD88] D. Chaum, C. Crepeau, and I. Damgård. Multiparty unconditional secure protocols. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM Press, 1988.

[CDD+99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT'99*, pages 311–326, 1999.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC: ACM Symposium on Theory of Computing – STOC'96*, 1996.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 383–395. IEEE Computer Society Press, 1985.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the Association for Computing Machinery*, 24(2):84–88, February 1981.

[Cha88] D. Chaum. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 2002.

[CKLS02] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

[CKPS01] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *LNCS*, pages 524–541. Springer-Verlag, 2001.

[CL05] J. Camenisch and A. Lysyanskaya. A formal treatment of onion routing. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 169–187. Springer-Verlag, August 2005.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2002.

[CM89] B. Chor and L. Moscovici. Solvability in asynchronous environments. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 1989.

[CR87] B. Chor and M. O. Rabin. Achieving independence in logarithmic number of rounds. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 260–268. ACM Press, 1987.

[CR93] R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience (extended abstract). In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 42–51, 1993.

[DDM03] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[DDN01] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, April 2001.

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, 1989.

[DM00] Y. Dodis and S. Micali. Parallel reducibility for information-theoretically secure computation. In *CRYPTO*, pages 74–92, 2000.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[DN00] I. Damgård and J.B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.

[DN03] I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO: Proceedings of Crypto*, 2003.

[DO00] S. Dolev and R. Ostrobsky. Xor-trees for efficient anonymous multicast and reception. *ACM Trans. Inf. Syst. Secur.*, 3(2):63–84, 2000.

[DSCP02] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop – PET ' 02*, volume 2482 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2002.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437, Los Angeles, California, 12–14 October 1987. IEEE.

[FLS99] U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SICOMP: SIAM Journal on Computing*, 29(1), 1999.

[FM85] P. Feldman and S. Micali. Byzantine agreement in constant expected time (and trusting no one). In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 267–276. IEEE Computer Society Press, 1985.

[FM88] P. Feldman and S. Micali. Optimal algorithms for byzantine agreement. In Richard Cole, editor, *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pages 148–161, Chicago, IL, May 1988. ACM Press.

[FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput*, 26(4):873–933, 1997.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology– AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, 1992.

[FS01] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[Fur04] J. Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. In *International Workshop on Practice and Theory in Public Key Cryptography – PKC ' 04*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[GBW98] I. Goldberg, M. Briceno, and D. Wagner. Gsm cloning, 1998. Available from `http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html`.

[Gen95] R. Gennaro. Achieving independence efficiently and securely. In *Proceedings of the 14th Annual ACM symposium on Principles of Distributed Computing*, pages 130–136. ACM Press, 1995.

[Gen00] R. Gennaro. A protocol to achieve independence in constant rounds. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):636–647, July 2000.

[GHPvR05] F.D. Garcia, I. Hasuo, W. Pieters, and P. van Rossum. Provable anonymity. In *Proceedings of the 3rd ACM Workshop on Formal Methods in Security Engineering – FMSE'05*, pages 63–72. ACM Press, November 2005.

[GJ04] P. Golle and A. Juels. Dining cryptographers revisited. In *Proceedings of Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*. Springer-Verlag, May 2004.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, pages 295–310, 1999.

[GK96] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput*, 25(1):169–192, 1996.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *Siam Journal of Computing*, 17(2):281–308, April 1988.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th Symposium on Foundations of Computer Science*, pages 174–187. IEEE, 1986.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, 1987.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991.

[Gol93] O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, 1993.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools (Appendix)*. Cambridge University Press, 2001.

[Gol04] O. Goldreich. *Foundations of Cryptography: Volume 2: Basic Applications*. Cambridge University Press, 2004.

[Gro03] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *International Workshop on Practice and Theory in Public Key Cryptography – PKC ' 03*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[Gro05] J. Groth. A verifiable secret shuffle of homomorphic encryptions. Full version of [Gro03], available from `http://www.brics.dk/~jg/JournalShuffle.ps`, 2005.

[GRR98] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing (PODC '98)*, pages 101–112. ACM Press, June 1998.

[GRS96] D.M. Goldschlag, M.G. Reed, and P.F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer-Verlag, May 1996.

[GT96] C. Gülcü and G. Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium – NDSS '96*, pages 2–16. IEEE Press, February 1996.

[HM97] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 25–34, Santa Barbara, California, 21–24 August 1997.

[HM00] M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.

[HM05] A. Hevia and D. Micciancio. Simultaneous Broadcast Revisited. In *the 24th annual ACM symposium on Principles of distributed computing PODC'05*, pages 324–333. ACM Press, 2005.

[HMq04] D. Hofheinz and J. Muller-quade. A synchronous model for multi-party computation and the incompleteness of oblivious transfer. Available from http://eprint.iacr.org/2004/016, 2004.

[HNP05] M. Hirt, J. B. Nielsen, and B. Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience (extended abstract). In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 322–340. Springer-Verlag, 2005.

[HO04] J.Y. Halpern and K.R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 2004.

[IKOS06] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. Cryptology ePrint Archive, Report 2006/084, 2006. `http://eprint.iacr.org/`.

[JJR02] M. Jakobsson, A. Juels, and R.L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium (SECURITY-02)*, pages 339–353, Berkeley, CA, USA, August 5–9 2002. USENIX Association.

[KEB98] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop – IH ' 98*, volume 1525 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[KSRW04] T. Kohno, A. Stubblefield, A.D. Rubin, and D.S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy*, pages 27–. IEEE Computer Society, 2004.

[LLM+01] M. Liskov, A. Lysyanskaya, S. Micali, L. Reyzin, and A. Smith. Mutually independent commitments. In *Advances in Cryptology – ASIACRYPT'01*, LNCS. Springer-Verlag, 2001.

[LLR02a] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated byzantine agreement. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 514–523. ACM Press, 2002.

[LLR02b] Y. Lindell, A. Lysyanskaya, and T. Rabin. Sequential composition of protocols without simultaneous termination. In *Proceedings of the twenty-first Annual Symposium on Principles of Distributed Computing (PODC-02)*, pages 203–212. ACM Press, 2002.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[Lys00] A. Lysyanskaya. Threshold cryptography secure against the adaptive adversary, concurrently. Report 2000/019, Cryptology ePrint Archive, 2000.

[MR90] S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *CRYPTO: Proceedings of Crypto*, 1990.

[MR91] S. Micali and P. Rogaway. Secure computation (abstract). In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer-Verlag, 1992, 11–15 August 1991.

[MR04] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer-Verlag, 2004.

[MRS88] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *Siam Journal of Computing*, 17(2):412–426, April 1988. Special issue on cryptography.

[MVdV04] S. Mauw, J.H.S. Verschuren, and E.P. de Vink. A formalization of anonymity and onion routing. In *European Symposium on Research in Computer Security – ESORICS ' 04*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[Nef01] A. Neff. A verifiable secret shuffle and its application to E-voting. In *SIGSAC: 8th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2001. Full version available from `http://www.votehere.net/vhti/documentation/egshuf.pdf`.

[Nie03] J. B. Nielsen. *On Protocol Security in the Cryptographic Model*. Ph.D. thesis, Aarhus University, 2003.

[NSNK04] L. Nguyen, R. Safavi-Naini, and K. Kurosawa. Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In *International Conference on Applied Cryptography and Network Security – ACNS'04*, volume 2 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[Ped91] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1992, 11–15 August 1991.

[PK01] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity — A proposal for terminology. *Lecture Notes in Computer Science*, 2009:1–9, 2001.

[PPW91] A. Pfitzmann, B. Pfitzmann, and M. Waidner. Isdnmixes: Untraceable communication with very small bandwidth overhead, 1991.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[PSW00] B. Pfitzmann, M. Schunter, and M. Waidner. Secure reactive systems. Research Report RZ 3206 (#93252), IBM Research, February 2000.

[PW87] A. Pfitzmann and M. Waidner. Networks without user observability. *Computers & Security*, 6(2):158–166, April 1987.

[PW00] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In Sushil Jajodia, editor, *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 245–254, Athens, Greece, November 2000. ACM Press.

[PW01] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In Francis M. Titsworth, editor, *Proceedings of the 2001 IEEE Symposium on Security and Privacy (S&P-01)*, pages 184–201, Los Alamitos, CA, May 14–16 2001. IEEE Computer Society.

[Rab94] T. Rabin. Robust sharing of secrets when the dealer is honest or cheating. *JACM: Journal of the ACM*, 41, 1994.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st Annual Symposium on Theory of Computing (STOC '89)*, pages 73–85, New York, May 1989. ACM Association for Computing Machinery.

[Rey01] L. Reyzin. *Zero-knowledge with public keys*. PhD thesis, MIT, 2001. Supervisor-Silvio Micali.

[RP04] M. Rennhard and B. Plattner. Practical anonymity for the masses with morphmix. In Ari Juels, editor, *Proceedings of Financial Cryptography – FC '04*, volume 3110 of *Lecture Notes in Computer Science*. Springer-Verlag, February 2004.

[RR98] M.K. Reiter and A.D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[RS91] C. Rackoff and D.R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992, 11–15 August 1991.

[RS93] C. Rackoff and D. R. Simon. Cryptographic defense against traffic analysis. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 672–681, May 1993.

[SA00] F. Stajano and R. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In Andreas Pfitzmann, editor, *Information Hiding —3rd International Workshop, IH'99*, volume 1768 of *Lecture Notes in Computer Science*. Springer-Verlag, October 2000.

[Sak00] K. Sako. An auction protocol which hides bids of losers. In *Public Key Cryptography – PKC'00*, volume 1751 of *Lecture Notes in Computer Science*, pages 422–432. Springer–Verlag, 2000.

[SD02] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop – PET ' 02*, volume 2482 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2002.

[Ser04] A. Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, June 2004.

[SG02] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. of Cryptology*, 15(2):75–96, 2002.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sho04] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[SP92] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[SS99] P.F. Syverson and S.G. Stubblebine. Group principals and the formalization of anonymity. In *Proceedings of the World Congress on Formal Methods (1)*, volume 1708 of *Lecture Notes in Computer Science*, pages 814–833. Springer, 1999.

[vABH03] L. von Ahn, A. Bortz, and N.J. Hopper. k-Anonymous message transmission. In Vijay Atluri and Peng Liu, editors, *Proceedings of the 10th ACM Conference on Computer and Communication Security – CCS'03*, pages 122–130, New York, October 27–30 2003. ACM Press.

[Wai90] M. Waidner. Unconditional sender and recipient untraceability in spite of active attacks. In J-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – EUROCRYPT ' 89*, volume 434 of *Lecture Notes in Computer Science*, pages 302–319. Springer-Verlag, 1990.

[Wik04] D. Wikström. A universally composable mix-net. In *Theory of Cryptography TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 317–335. Springer-Verlag, Feb 2004.

[WP89] M. Waidner and B. Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology–EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, page 690. Springer-Verlag, 1990, 10–13 April 1989.

[Yao86] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, 1986.