Provably Correct, Secrecy Preserving Computation and its Applications in Auctions and Securities Exchanges

A dissertation presented

by

Christopher Andrew Thorpe

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

May 2008

©2008 - Christopher Andrew Thorpe

All rights reserved.

Dissertation Advisors: Author

David C. Parkes and Michael O. Rabin **Christopher Andrew Thorpe**

Provably Correct, Secrecy Preserving Computation and its Applications in Auctions and Securities Exchanges

Abstract

Recent advances in cryptography provide powerful new tools for enhancing trust in electronic commerce at low cost. We construct a general model of provably correct, secrecy preserving computation without relying on any particular cryptographic framework or assumptions. This model employs an "Evaluator-Prover" that accepts encrypted inputs from many (possibly unaffiliated) parties, computes one or more functions on those inputs, outputs the functions' results and verifies the correctness of the results to one or more verifiers. We distinguish our work from other secure computation approaches as a balance between absolute security and a completely trusted third party, achieving a model enjoying computational tractability and suitability for business applications.

Our evaluator-prover is not trusted in the traditional sense; it is bound to output only the correct results at all times and prevented from disclosing private data by tools from other areas of computer science research such as trusted computing and network security, rather than the provably secure cryptographic tools employed in many past solutions. We show how to construct an implementation of our model using Paillier's homomorphic encryption scheme. We propose a "time-lapse cryp-

tography service" that produces public encryption keys and guarantees decryption at a particular time by constructing and releasing the corresponding decryption key after a specific interval. This service functions as a new cryptographic commitment primitive with binding, hiding, and nonrepudiation.

Provided with these tools, we construct four new mechanisms for electronic commerce: a cryptographic sealed-bid auction protocol for one or more identical items, a cryptographic combinatorial auction protocol based on the "clock-proxy" auction, a cryptographic securities exchange that conducts a continuous double auction for a particular security, and a cryptographic combinatorial securities exchange that provides for efficient atomic exchange of baskets of many securities.

Along the way, we develop useful building blocks of independent interest, most notably a novel cryptographic mechanism to efficiently prove a solution to a linear or integer program is optimal based on its encrypted inputs and encrypted constraints; this provides unprecedented efficiency in proving the correctness of winner and price determination in our combinatorial clock-proxy auction.

Contents

Citations to Previously Published Work

Large portions of Chapters 5 and 6, as well as some of Chapter 3, appeared in the following conference paper [115] and will appear in the following journal article [119]:

> "Practical Secrecy-Preserving, Verifiably Correct and Trustworthy Auctions", David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher A. Thorpe, *Proceedings of the 8th International Conference on Electronic Commerce*, ACM Press, pp. 70–81, 2006; and to appear in *Electronic Commerce Research and Applications*, summer 2008.

A conference paper version of Chapter 8 [150] appeared in:

> "Cryptographic Securities Exchanges", Christopher Thorpe and David C. Parkes, *Proceedings of Financial Cryptography and Data Security*, Springer, pp. 163–178, 2008.

Chapter 4 was published as a Harvard SEAS technical report [129]:

> "Time-Lapse Cryptography", Michael O. Rabin and Christopher Thorpe, Technical Report TR-22-06, Harvard University SEAS, 2006.

Acknowledgments

Writing acknowledgments is one the most nerve-wracking parts of the dissertation. One always feels that one risks inadvertently forgetting someone, particularly with a set of acknowledgments this long. However, since a Ph.D. as the culmination of decades of education, it feels appropriate to do my best to thank those who contributed to my education.

First, I thank the many teachers in my elementary and secondary education, and the professors, lecturers, preceptors and teaching fellows in the dozens of classes I have taken at Harvard and MIT as an undergraduate and graduate student.

A Harvard education comes not only from the faculty, but also from the students and other members of the Harvard community who learn from and teach one another. Those among my undergraduate and graduate years who I remember well for helping me, many of whom remain friends, include the following; there were many others who also helped me whom I don't remember here but my imperfect memory does not diminish their contributions. Ben Adida, Attila Bódis, Bonnie Berger, Eliza Block, Olivia Brown, Guillermo Diez-Cañas, Julia Carey, Mark Catrell, Benson Chang, Tom Cheatham, Brad Chen, Hamilton Chong, Etan Cohen, Jonas Cooper, Adam Deaton, Aram Demirjian, Justin Deri, João Dias, Ying Du, Sofía Echegaray, Rachel Elkin, Dan Ellard, Julie Farago, Allan Friedman, Kathy Gerlach, Robin Goldstein, Josh Goodman, Paul Gosmorino III, Paul Govereau, Jon Grenzke, Dick Gross, Robert Haas, Michael Hammer, Jessica Hammer, Jan Hanseth, Mike Hlavač, Dan Horwitz, Luke Hunsberger, Andrew Jacobs, Jason Jay, Ece Kamar, Emir Kapanci, Harry Kargman, Brian Kernighan, Adam Klivans, Catalina Laserna, Lillian Lee, Brad Leupen, Alex Lewin, Austin Lin, Sean Lyndersay, Pia Malkani, Phil Maymin, David

Mazières, Clayton Myers, Alexander Nadolinski, Brenda Ng, Brian Park, Kathy Park, Lori Park, Anand Pathak, Annelisa Pedersen, Andrew Pimlott, Heather Pon-Barry, Kevin Redwine, Ren Richmond, Jonathan Roberts, Nailah Robinson, Floris Roelofsen, David Ryu, Peg Schafer, Sven Seuken, Chung-chieh ("Ken") Shan, abhi shelat, Eric Silberstein, Janis Siggard, Chris and Kate Sims, Carl Sjogreen, Matt Tambiah, Dylan Thurston, Jesse Tov, Omri Traub, Mike Tucker, Emmanuel Turquin, Bob Walton, Ilan Wapinski, Dan Williams, Elif Yamangil, Eric Yeh, Cliff Young, Ruth Zeltzer, and Mark Zuckerberg.

The students in my CS286r class this semester taught me at least as much as I taught them, and they were all extremely understanding about how busy I was. I am also indebted to my exceptional teaching fellows, Olivier Guéant and Tal Levy.

The administrative staff in SEAS who support our research groups deserve special appreciation for taking care of all sorts of important things that faculty and graduate students are often not very good at. In particular, Arthur Cram, Hetchen Ehrenfeld, Sandi Godfrey, Carol Harlow and Tricia Ryan have been especially helpful.

I once mentioned to Susan Wieczorek that she must be the most frequently acknowledged individual among SEAS Ph.D. dissertations. When I expressed surprise when she responded that she has not been acknowledged in every one of them, she wryly retorted, "Not everyone is as much trouble as you are." She's undoubtedly right, for between taking a year off, switching advisors to a pair of faculty, scrounging for funding, defending my thesis with one advisor in Israel, and getting everything in order for my class this spring, I can't think of a more apt word than "trouble." We are fortunate to have someone with her administrative acumen keeping so many

students out of trouble—and out the door. In over a decade, I have yet to her a question for which she didn't either know the answer or the right person to ask.

A few individuals who have contributed to this research, whether directly or indirectly, merit particular recognition.

I thank Zulfikar Ahmed, Michael Barrientos, Jonas Budris, William Josephson, Stephanie Lee, Andrew McCollum, Nina Ni, and Greg Valiant, for their understanding of my obligations as a student and their hard work on our joint endeavors.

Chris Small has been a good friend and colleague since he was my CS50 TF; he was especially helpful in understanding the systems implications of time-lapse cryptography and commercial interest in our research.

Alon Rosen cheerfully answers many of my questions on the literature in the field of cryptography.

Carlos Battagglia taught me a few valuable guiding principles you're more likely to learn while fishing in Texas than sitting in a classroom at Harvard.

Aggelos Kiayias pointed me to Boudot's paper on efficient interval proofs [30].

Lee Fleming, Mitch Hesley and Brian McRoskey contributed hard work and insight in commercial applications of these technologies in the context of Prof. Fleming's course "Commercializing Science and High Technology" at Harvard Business School.

Eric Budish deserves acknowledgment for his willingness to discuss my research applications in finance, his useful references to relevant literature, and a number of helpful insights that have influenced our work and its presentation – particularly in a way that is comprehensible and motivating to the finance community.

Imad Labban and Stephanie Borynack of Lehman Bros., who helped to understand

the utility of our work in a business context and inspired the problems we solve in Chapter 9.

Eric Allman's friendship, kindness and encouragement have been a source of inspiration through this process.

Kirk McKusick gave me practical advice about a Ph.D. with a candor that one rarely finds among career academics. "It doesn't have to be your life's work," he declared when I asked him for advice about picking this project two and a half years ago; just find a project, prove that you are able to do real research, and get it done.

Román Román has always been there for me.

Anthony Johnson helped to keep me sane through this process, especially through forcing me to have fun when I was getting too stressed out.[1]

If I had never studied computer science, I never would have taught Jennifer 8. Lee in CS51 her freshman year at Harvard. We have been close ever since. Her first book was just published in March 2008, and our mutual support on these large projects has been a real joy.

I thank the other faculty who have been so warmly supportive of our work, including among others Steven Gortler, Barbara Grosz, Tom Kelly, David Malan, Michael Mitzenmacher, Ron Rivest, Tuomas Sandholm, Leslie Valiant, and Matt Welsh.

Stuart Shieber made important contributions to our earliest joint work in the field and provided important referrals to industry contacts, and ongoing encouragement and support.

Salil Vadhan shared his deep insight into the theoretical fundamentals of cryptog-

[1]For example, insisting I find a way to integrate the word "nuffie" into this document.

raphy and provided helpful answers to many questions.

Bob Kendrick taught me by example how to love early music and how to teach in a way that instills a love of the subject among the students in a class.

Harry R. Lewis called me on the phone and convinced me why Harvard was a better choice than MIT when I was a senior in high school. His teaching and excellent advice on many matters over many years have been an important part of my experience here. His love of and service to Harvard are truly unparalleled.

John Y. Campbell invited me to present my research at the weekly Finance Lunch and I thank him for his hospitality when I reached out to the finance community at Harvard, and for asking hard questions that led to important insights in our work.

Rocco Servedio has been a collaborator and friend since my sophomore year of college, when we both took Computer Science 121 from Harry Lewis and attended sections led by then undergraduate, and now full professor, Salil Vadhan. After having been partners in Harvard's infamous undergraduate operating system class, collaborating on theoretical research with him is a piece of cake.

I thank Henry Leitner, for a long and productive relationship teaching CS51, the second semester introductory computer science course, and for offering me numerous opportunities to grow through course development and leadership. I count my time teaching with Henry as some of the most meaningful – and educational – at Harvard.

I wish to thank Norman Ramsey, from whom I have learned so many useful things. While the grace and lucidity of his prose and the courtesy of his communications continue to delight me, his truly selfless love of his students, the art of pedagogy, and the joy he takes therein are among the chief inspirations that have led me to leave

one foot in the door of academia.

Before thanking my undergraduate thesis advisor and committee, I wish to thank all the members of my family who have helped me along the way to become who I am, and four in particular with respect to this work.

My uncle, Andy Patrick, who encouraged me to finish the Ph.D. and not drop out to do a startup. "There are always startup and business opportunities, but a Ph.D. from Harvard is really rare." He was right, and I'm glad I followed his advice.

My grandfather, Rodger Patrick, spearheaded the adoption of computers at Hewitt Associates five decades ago to solve enormous actuarial problems. I am proud to continue a family tradition of advancing the adoption of computing technologies in business. He has always been a great support and inspiration to me; I have always admired his work ethic, intelligence, and programming skills—even I would be hard pressed to write a Sudoku solver in Microsoft Excel.

My parents, Allen and Elizabeth Thorpe, have two children, and my brother and I are on the way to earning a total of five graduate degrees. I am grateful to them not only for teaching me the importance of education, but also giving me the freedom to make a lot of mistakes and learn from their consequences. If parents managed to create a perfect world for their children, how unprepared would their children be for real life?

The late David Lewin, of the Harvard Music Department, inspired me to pursue teaching by showing me the incomparable love for a field of study a teacher can instill in his students. He gave me the confidence to successfully pursue a joint undergraduate concentration in computer science and music, and to begin a Ph.D. in the same area. He passed away within the week of teaching his last class before his planned retirement from Harvard in May 2003, and we miss him.

I thank Luis Viceira for generously contributing his time and ideas to this research. His expertise in finance has been invaluable, and his fundamental insights about the economic value of trust have influenced our own understanding of the impact of our research. I am especially grateful to him for his participation on my thesis committee.

Margo Seltzer, who taught CS50 my freshman year as a Harvard undergraduate, hired me – not even eighteen years old – as a TF for the following year. Margo's excellent teaching, love of the material, and generous attention were instrumental in my choice of computer science as a field of study. While I have always been grateful for her open door, warm smile, and sage advice throughout the years I have had the pleasure to learn from her, I am most recently indebted to her for her willingness to serve on my committee at the last minute.

Avi Pfeffer was my first graduate advisor at Harvard when I first returned to study applications of artificial intelligence in music. Avi and I still share a strong interest in research in that area. Although I have ultimately completed the Ph.D. in a completely different area than we expected, I am grateful for his continued friendship and support, and for his valuable feedback in framing our research in a way that is comprehensible and motivating to a general computer science audience.

David Parkes has been a wonderful colleague and valuable friend since our earliest research collaboration. Rarely have I found myself so productive as when David and I tackle interesting problems; it often feels like the papers write themselves when our complementary skills leave few open questions to worry about. I'm also truly grateful for the significant investment of time he made in my development and our joint research. With a number of Ph.D. candidates in the pipeline, he has unselfishly made sufficient time for us to work together, and delivered lucid drafts of complex ideas at breakneck speed. I look forward to a long collaboration with David in cryptographic computational mechanism design, and I am particularly grateful for the opportunity to teach his graduate course this past semester. I heartily congratulate him on his recent promotion to the rank of Professor with tenure.

Michael Rabin, who has been hailed by greater minds than mine for a singularly productive career, has given me the confidence to achieve goals I would never have thought possible on my own – and indeed, the achievements in this work would never have been possible without his encouragement and ideas. It has been a tremendously rewarding experience to collaborate with him on the theoretical cryptographic tools supporting this work, and I look forward to our upcoming extensions to the RST cryptosystem. Working with Prof. Rabin is also a lot of fun. We both share a love of wordplay, and it is in keeping with his sense of humor to end this section on that note. My favorite recollection of his wit is from over a decade ago, when as an undergraduate I watched him write values $b_1, \ldots, b_{12}$ on the blackboard and remark, "Remember, b_{12} is a coefficient; it's not a vitamin, so don't eat it." I'm truly grateful for his support, friendship, and TLC over the past three years.

For M. R., D. P., A. P., and Δ*.* Λ*.,*
without whose generosity of wisdom
this never would have been.

Preface

Students should keep in mind that GSAS and many departments deplore overlong and wordy dissertations.
—The Form of the PhD Dissertation, Harvard GSAS

Therefore, since brevity is the soul of wit,
And tediousness the limbs and outward flourishes,
I will be brief:
—*Hamlet*, Act I, scene ii.

While not every dissertation has or needs a backstory, this particular dissertation rather seemed to just happen. One need not know this story to appreciate its contributions, but I hope that the curious history of this nearly accidental dissertation may be of interest, and serve to "connect the dots" between its various chapters—and why a student who entered the Ph.D. program to study applications of artificial intelligence in music graduated with a dissertation on new cryptographic techniques and their applications in electronic commerce.

When I was still in elementary school, I was one of those kids who enjoyed making up and sharing "secret codes" with my friends; we'd transmit messages to each other encrypted with a simple Cæsar or substitution cipher. Even though the information could not have been of great import, decoding them was more than half the fun; watching the secret message unfold was like realizing what's underneath the wrapping paper while opening a birthday present.

So when I decided to return to graduate school to finish a Ph.D. in the intersection of artificial intelligence and music after a year on leave, I noticed Prof. Rabin was teaching his cryptography course, and I still had a theory requirement to complete. I had taken his course on randomized algorithms ten years before as an undergraduate,

and it was (with all due respect to other faculty) the theoretical course I found most interesting in my undergraduate curriculum. So I knew it would be well taught, and I was curious to understand what real cryptography – not just the stuff of schoolchildren and Neal Stephenson novels – was all about.

A little more than halfway through the semester, no one was more surprised than I was when one day, sitting in the lecture hall before crypto class, I hear a voice behind me – "Chris!" I turned around, and Prof. Rabin had just entered the room and was smiling at me. The conversation went something like this:

"Hello, professor!" I stammered. I was surprised that a member of the theory faculty had anything to ask me – after all, I thought of myself as kind of a systems guy who could get by in AI, if only to code up techniques other people came up with to solve problems I thought interesting.

"Stuart Shieber and David Parkes and I have been working on a project and I would like to talk to you about it. Can you meet with me to discuss it?"

That week, Prof. Rabin outlined a new approach to sealed-bid auctions that he and Profs. Shieber and Parkes were working on. Shieber had approached Rabin with the idea of a cryptographic auction, and they soon invited Parkes, an expert in computational mechanism design, to join the conversation. They still wanted another member of the team: someone who could implement the scheme and evaluate its efficiency and thus its feasibility for real-life applications in business, and apparently my name had come up in their conversation.

This dissertation owes its existence to this happy sequence of events, one of those lucky conjunctions of people and ideas where a spark of a simple idea lights in-

teresting new research. This project eventually turned out to be the holy grail of graduate school: a research topic more interesting than any of one's preferred methods of procrastination. I found myself staying up past midnight, not to vanquish computer-generated dragons, but to figure out how to efficiently implement the necessary cryptographic computations and extend the professors' original approach to more general auctions of multiple identical items with strategyproof pricing models.

Before this project, I knew precious little about auctions, or even computational mechanism design, and my formal cryptography background was largely limited to the content of Prof. Rabin's graduate course. While the cryptography required for conducting a sealed-bid auction was no more complex than what we studied for electronic voting in class, Prof. Parkes was instrumental in teaching me enough about auctions to make a dent in the problem. He knew exactly what to tell me to read and patiently answered my many questions, and our collaboration soon bore fruit.

Within a few short months, and a number of long meetings, we had our first paper complete; it was soon accepted to a conference on electronic commerce and one of six papers at that conference selected to appear in a special issue of the journal *Electronic Commerce Research and Applications,* and is incorporated into the present work as Chapter 6.

One thorny issue in that work was the problem of repudiation: while we could employ binding and hiding cryptographic commitments to keep bidders' bids safe from the auctioneer before the auction closed, and prevent them from changing their bid after bidding, we couldn't prevent a bidder from refusing to "unlock" a commitment to a bid. In some high-stakes auctions, this could be a real problem: if I realize that

I significantly overbid, it might well be in my best interest to just refuse to open my commitment altogether, and no one will be able to hold me to it.

We discussed various solutions. Parkes, who has significant experience in real-life high-stakes auctions, suggested a fine for failing to open a commitment. He admitted that this was not particularly attractive, as setting the amount of the fine correctly requires an understanding of the value at stake – something that is often unknown until the auction discovers an item's value. Thus, it seemed hard to make a fine an appropriate deterrent without making it impractical. Distributing bid information among other bidders with threshold secret sharing complicated the protocol significantly and placed an undue level of trust in other bidders. Using a trusted third party was, as usual, possible but unappealing because the third party now knows secret information, so we thought about encrypting the information with a public key in a way that it couldn't be decrypted until after the auction.

This idea of Prof. Rabin's was first described as a "delayed private-key revelation service": by having a widely known public encryption key to encrypt, but keeping the decryption key secret until later, one could ensure the secrecy of the bids *during* the auction but guarantee that they could be available to the auctioneer *after* all bids had been submitted. While we alluded to such a service in our first paper, we did not fully formalize it.

At that point, Prof. Rabin pointed out that a set of third parties trusted not to collude could offer a service which published public encryption keys at one point in time, and delayed publication of the secret decryption keys until later. That service could be employed to force nonrepudiation of bids, and be completely separate from

the auction infrastructure. Bids would be encrypted with a public key generated by a service, and if a bidder refused to open a bid, it could be forced open when the service published the secret key.[2]

We sketched such a service in the auctions paper, and after our work on auctions was submitted and accepted, Prof. Rabin and I set out to formalize such a service. When trying to describe the name, I coined the phrase "time-lapse cryptography" (intentionally echoing "time-lapse photography", though the meaning of "time-lapse" admittedly differs) and it stuck. After we spoke for a few minutes about building distributed ElGamal keys as done in his class, I set out to think about it, armed with my lecture notes from his course and a good search engine. We agreed to meet, and I sent him a draft of my solution the night before.

He confessed he hadn't had time to look at my draft and first wanted to present his solution. As his skilled hands sketched on the whiteboard the mechanism he had in mind, I was delighted that we had come up with exactly the same protocol – including both of us independently choosing Feldman's verifiable secret sharing [61] to guarantee correspondence between the public and private keys. After a few very productive meetings, we completed a technical report which now forms the basis for Chapter 4 of this work.

We are now looking forward to impementing a prototype "Time-Lapse Cryptography Service" that will publish vast numbers of cryptographic keys anyone can employ and use in any number of cryptographic applications.

[2]When preparing a technical report on this paper, we found that the concept of a delayed key revelation service, and its application to auctions, had already been considered independently; see Chapter 4 for details.

At about the same time as Prof. Rabin and I were completing that work, I had still been thinking about other applications for the auctions research. Originally, we had considered an application to bond auctions, but online "direct to retail" bond auctions had a discouraging track record, even though they seemed to benefit the corporations issuing the bonds.[3]

So, while bond auctions seemed attractive, they also seemed out of reach from a business perspective. I began to think about other auctions in securities; having done a good deal of equities trading in my private life it occurred to me that perhaps our ideas could be applied to securities exchanges, where information about the orders could be hidden just like bids in a sealed-bid auction.

After I presented a sketch of this idea to Prof. Parkes, he obtained a number of market microstructure references from his colleagues at Harvard Business School which I devoured. After some initial forays into the literature, it became clear that exploitation of order information in securities exchanges does lead to very real costs to traders; the most common, front-running and penny-jumping, are too often practiced (and, thankfully, prosecuted).

Parkes and I rapidly assembled a paper on how to run a continuous double auction of securities using a homomorphic cryptosystem similar to that we used in sealed-bid auctions. In our system, a partially trusted market operator proves which trades should happen without revealing the magnitude of the trades – or even the exact prices – to other market participants. Thus traders might be more inclined to place

[3]This was primarily because because investment banks and their clients make a great deal of money underwriting these primary market bond auctions to very large bidders, who then "chop up" the bonds and sell them (typically for a profit) in a secondary market. Underwriting a bond auction directly to retail investors cut out a middleman who might take its other investment banking business elsewhere if it no longer had relatively exclusive access to lucrative primary offerings.

limit orders, knowing that the information in them would not be exploited (or, at least, exploited less, depending on how much information is revealed). This paper appeared in the 2007 *Financial Cryptography and Data Security* conference and forms the basis of Chapter 8.

After Prof. Rabin returned in the fall, we began to meet occasionally to discuss our work, and future research and commercial applications of it. One morning, he called me on my mobile phone, audibly excited with an idea, soon to be published in 2007 by Rabin, Servedio and Thorpe; I will refer to it as the "RST" cryptosystem for brevity.[4] This work was primarily motivated by an application to sealed-bid auctions, in order to both reduce the computational burden of homomorphic cryptography as well as reduce dependencies on cryptographic security assumptions – only secure cryptographic commitments are required.

A few minutes after the phone rang, I sat before a whiteboard, feeling rather like an attendee in a private lecture, watching and asking question after question, as he presented a sketch of a new cryptosystem in which encryption was done by placing the message x in a finite field F_p, then choosing two random numbers $\langle x_1, x_2 \rangle$, such that they sum to the message (modulo the size of the field): $x_1 + x_2 \equiv x (\text{mod } p)$. By committing to both numbers, one has "encrypted" the message – revealing both commitments reveals the message by summing them. But if one number is revealed, then nothing is known about the message with information theoretical security.[5]

Then he demonstrated how if you have two numbers represented in this way, say, $x : \langle x_1, x_2 \rangle$ and $y : \langle y_1, y_2 \rangle$, then $\langle x_1 + y_1, x_2 + y_2 \rangle$ represents, naturally, $x + y$ – the

[4] Cryptographers whose last names begin with Q or U are cordially invited to join our project.
[5] Provided, of course, that the commitment scheme is also information-theoretically hiding.

sum of all four coefficients. Then, if I ask you to reveal only the first or second "half" of each number and the sum, then I've learned nothing: I've learned two random values and their sum. Only if I learn both halves do I learn anything. But, if you offer me twenty such representations of $x, y, x + y$, and I ask you to reveal half of nineteen of them – where I pick which half – then I will detect any cheating, i.e. the twentieth sum is not really the sum of the first two values, with astonishingly high probability. Given such a protocol for provably correct addition, it is possible to construct a similar protocol for provably correct multiplication. Only a bit more was needed to use this system for proving a sealed-bid auction correct.

The next challenge was proving correct interval proofs in this context. After my presentation at *Financial Cryptography* on cryptographic securities exchanges, Aggelos Kiayias had pointed out that there was an interesting body of recent work on interval proofs that improved upon the cut-and-choose methods we employed in our early work; he later pointed me to Fabrice Boudot's paper [30] on the subject. I followed up these leads and presented them to Prof. Rabin, who constructed a fascinating interval proof for values encoded in this protocol using the fact that every integer can be represented as the sum of four squares, proved by Lagrange in 1770.

It is one of those mysterious and beautiful coincidences of science that two hundred years after Lagrange's proof, in the 1970's, Rabin himself devised an efficient probabilistic algorithm to find four squares that sum to a particular integer; he and Shallit published this algorithm in 1986.[6] Twenty years after that, Rabin discovered a new important use for his own number-theoretic algortihm: interval proofs in the

[6]Another such coincidence is that Joseph Lagrange himself was Prof. Rabin's advisor's advisor's advisor's advisor's advisor's advisor's advisor [3].

RST cryptosystem. When running experiments on the efficiency of the RST cryptosystem, I downloaded Peter Schorn's Python implementation of the Rabin-Shallit algorithm, which was so blindingly fast that that computation was comparatively negligible.

Soon after our initial discussions, Prof. Rabin traveled to New York to teach at Columbia for a semester, where our friend Prof. Rocco Servedio is on the faculty. We soon completed a general protocol for secrecy-preserving, provably correct straight-line computations, and we showed in initial experimental results that using this new method for sealed-bid auctions was considerably more efficient than our previous work using Paillier encryption.

Prof. Rabin presented this work, "Highly Efficient Secrecy-Preserving Proofs of Correctness of Computations and Applications", at the 22nd IEEE Symposium on Logic in Computer Science last year, and we are presently working on a journal version of the article. While this work also supports the general framework of computation described in Chapter 2, we omit the work from this dissertation and refer the interested reader to the conference proceedings for details.

In the spring of 2007, I completed my Ph.D. course requirements with a class at Harvard Business School entitled "Commercializing Science and High Technology", taught by Lee Fleming. Fleming's course involved both close readings of business cases involving, well, commercializing science and high technology, and real-world, hands-on work in commercializing scientific research at Harvard. David Parkes and I co-sponsored our research on cryptographic securities exchanges. Mitch Hesley, Brian McRoskey and I formed a team where we investigated applications of this technology

in real finance.

A few interesting ideas emerged from the study of the course. The first was that because of new SEC regulations (known as "Reg NMS"), using a system such as ours for establishing prices may now not be possible on US-regulated cash traded equities exchanges – though exchanges in other jurisdictions and other types of securities might still benefit from our technology.

Another was that the original work we had done on cryptographic auctions had more application in commercial procurement than we had anticipated. While we have not yet pursued studying our work in procurement auctions in any formal way, this may be a very interesting and important area of further research.

Probably the most useful contributions of the course came at its end, when industry professionals from all over the Northeast came to a poster session describing our technology and our projects. Stephanie Borynack, responsible for University Relations at Lehman Bros., set up a meeting with myself and their director of program trading, Imad Labban. Mr. Labban and I discussed my technology briefly, and its application in so-called "dark pools" (where liquidity for block trades can be found without revealing your own liquidity unnecessarily). While our research sparked interest, he agreed that there were still a lot of problems to think about.

Mr. Labban then described to me a type of commercial transaction in which an institution with significant assets wishes to trade a very large basket of securities, worth hundreds of millions of dollars or more. In these transactions, the investment banks "buying" the baskets need to understand details of the risks of these baskets without being able to learn the exact equity positions that comprise them. (Naturally,

if anyone learned the exact positions being traded before the trade was agreed on, that information could be exploited, and even the most respected investment banks have found that, on occasion, an unscrupulous employee cannot resist the temptation of exploiting knowledge.) Could cryptography improve price discovery in these transactions by allowing the banks to better estimate their risk, without revealing any more information about the baskets?

This question led to the development of two new protocols for securities trading that may well be of interest in other commercial transactions. The first protocol attempts to solve the problem posed directly, by providing a mechanism by which the "buyer" can indirectly measure the change in its risk parameters by accepting an incoming good without learning anything about that good not implied by the risk changes.[7] This could improve price discovery in the securities transactions, because the banks could more accurately assess their actual risk by taking on a basket than before.

The second protocol combined ideas from our earlier cryptographic securities exchange work with this notion for a new protocol for a cryptographic alternative trading system for block trades. In this system, trading interest from many parties is encrypted and combined into a "cryptographic dark pool", and orders "cross" when one party wants to buy and another wants to sell. In many existing pools, if a trader wants to buy more than others are willing to sell, the order doesn't get filled; this makes such pools less attractive, and unusable when execution is essential.

Our protocol solves this problem by allowing the pool to generate an encrypted

[7]For example, if the function defining the feature is linear, then knowing the change in the feature of the buyer's risk implies knowing the feature for the good.

"remainder" basket of unfilled interest, whose risk characteristics can be proven to investment banks, and the most competitive bank can fill the unfilled liquidity, thus helping the parties to reach equilibrium. The banks then can accurately measure their risk in accepting the baskets, and the traders receive better prices because they can trade with all interested parties at once, not simply via pairwise trades. These protocols are outlined in Chapter 9.

The most recent contribution of the present work stems from David Parkes' observation that our existing mechanisms should be extensible to combinatorial auctions, where buyers place bids on bundles of many different goods. Parkes pointed out that the ability to efficiently conduct cryptographic combinatorial auctions would be a significant contribution, given the complexity of computing an optimal result. We have accordingly spent a good amount of time thinking together about how to apply the cryptographic tools we now have to these auctions. This part of the work, while most recently completed, has been the most elusive, primarily due to the complexity of the combinatorial auction problem.

Parkes' solid understanding of the cryptographic primitives, coupled with his world-class expertise in computational mechanism design, led him to suggest an ideal mechanism in which the cryptographic primitives can be applied to solve an important type of combinatorial auction: the combinatorial clock proxy auction. His contributions to this area of our research are tremendous, and the

In combinatorial auctions, a seller wishes to auction off a number of different goods, and a number of buyers wish to buy one or more of these goods in bundles. Once each buyer states its value on particular bundles, the problem of computing

an optimal allocation of items to buyers is computationally extremely hard. Thus, earlier techniques described in related work (cited in Chapter 7), such as multi-party computation of the results, or homomorphic cryptography involving enumeration of all possible allocations, are intractable in practice.

Parkes explained that if we could prove the solution to an integer program to be feasible, then a third party could perform the complicated computation of calculating a feasible solution, and the more expensive secrecy-preserving computations could be employed in the more efficient cases of validating the proofs of feasibility, given the same inputs. This fits extremely well with the "partially trusted third party" models of our other research, where we employ a third party to perform expensive computation and prove its actions correct, but do not trust that third party with any private information until after it cannot benefit from exploiting it.

Parkes further identified the "combinatorial clock" auction as an ideal candidate for the application of cryptography. In the clock auction, the auctioneer raises prices in a sequence of rounds while bidders indicate how many of each good they want at those prices. prices of goods until When the demand does not exceeds the supply of any item, the clock phase concludes with the current round's prices set.

Then, a final "proxy" sealed-bid auction is employed to eliminate possible abuse of the protocol and improve accuracy of reported prices. Employing the cryptographic protocols underlying our work, bidders can prove they obey activity rules that promote honest bidding without the cooperation of the auctioneer, and can jointly compute aggregate demand without revealing individual demand. Prof. Rabin suggested several optimizations to our initial ideas that make these computations intuitive and

efficient. We maintained our desired level of trust: only after all final bids are submitted does the auctioneer learn anything; he cannot influence the bidding with any private information.

This auction setting led to one of the most important technical contributions of the present work: the efficient yet provably correct secure computation of mixed-integer and linear programs. These optimization problems are well known as being among the most computationally expensive to solve, and thus elude secure computation via traditional, more theoretically secure means. Our models lend themselves very well to solving these optimization problems, because our third party can compute a correct result using high performance computation—which may still be exponential in cost—and then issue a proof that can be checked in polynomial time over the encrypted inputs, using slower verifiable computation techniques. These proofs were the last piece needed for our cryptographic combinatorial clock-proxy auctions described in Chapter 7.

Chapter 1

Introduction

A valuable technology lowers the economic costs of creating a good or providing a service, increasing the efficiency of a market. In this work, we develop new cryptographic technologies that reduce the cost of establishing trust in commercial and financial entities, by providing them with tools that serve two main purposes: proving that they performed all actions according to their obligations, and controlling access to any private information to limit its exploitation. We formulate a set of basic cryptographic tools, describe how to build these tools upon theoretical foundations of cryptography, and then use these tools to design computational mechanisms for trustworthily conducting auctions and trading securities.

The singular theme of our contribution is controlling trust with efficient, realistic cryptographic and computational tools. We seek to eliminate a large degree of trust by forcing every party to prove that its actions are correct—without revealing new information about the inputs leading to its actions. We further eliminate trust by protocols that bind participants to their commitments, even if they abandon the

protocol. And, where we cannot completely eliminate trust, we contain it by delaying private information from reaching any party entrusted with that information until it is essential; this typically reduces any benefit from breaching that trust.

Of course, cryptography itself has already become an essential component of modern electronic commerce: encryption protects against unauthorized access to private information, passwords and trapdoor functions provide authentication, and digital signatures and message authentication codes prove a message arrived unaltered from a known entity. As electronic commercial mechanisms have become more and more sophisticated, new cryptographic techniques of equal complexity have arisen to support them.

Yet there is a large class of commercial problems for which recent advances in cryptography are untapped, even though these advances offer solutions that are at once surprising or even seem impossible. For example, one can conduct a sealed-bid auction where the bids are kept secret until the end of the auction, the outcome can be proven correct without revealing any information not implied by the outcome itself, and bidders cannot repudiate their bids after they are submitted. Another example is an alternative trading system that can offer investors the opportunity to trade baskets of securities with each other without revealing the securities comprising those baskets – the system reveals only the risks associated with accepting a proposed trade.

In these real-world situations, practical cryptographic solutions are essential. While significant existing cryptographic and theoretical computer science research offers strong, information-theoretically secure solutions that enable such secure computation, these strong solutions come at a significant complexity cost, requiring dis-

tributed trust and the cooperation of multiple parties (among whom that trust is distributed) to securely compute results. Moreover, although such multi-party computation (MPC) can compute established trust of these trusted third parties, and there are no practical computational mechanisms for many large-scale settings. In real world examples, auctioneers and procurement officers are trusted not to reveal private information about bids before winners are determined, and to behave ethically. Trading firms are expected not to disclose clients' intended trades. Today, reputation plays a crucial role in trust: commerce relies on a vast number of entities whose trust is based almost entirely on their reputation. Despite a common reliance on reputation, there is ample evidence that these trusted third parties often should not be trusted: fraud often happens. Chapters 5 and 8 offer a number of documented examples of fraud in auctions and securities exchanges, respectively.

This fraud can come at tremendous cost. The unethical cooperation between a few senior auditors at Arthur Andersen and Enron executives led to an obstruction of justice conviction that destroyed Andersen's reputation, leading to the destruction of its business—even though it was one of the highly respected "Big Five" accounting firms [57]. A decade earlier, after two managing directors at Salomon Bros. were discovered to have placed bids on behalf of nonexistent clients in Treasury bond auctions to manipulate bond prices, the scandal eventually led to the resignation of Salomon's chairman and CEO who knew about the scheme but failed to inform the US Government. The event cost Salomon $290 million in fines and penalties, forced its chairman to resign, and tarnished its reputation for years [56]. More recently, the ultimate effects of widespread fraud and lax compliance in subprime mortgage

lending [9, 18] began a chain reaction whose effects have included the near-destruction of financial giant Bear Stearns, a crisis in the credit markets and associated volatility in world markets, the inability of borrowers to obtain mortgages and sharp declines in real estate prices.

These events are clear evidence of the tremendous economic value of trust in our commercial and financial institutions. Building an institution that merits trust through reputation is extremely costly and can take many years—while a breach of that trust can destroy a trusted institution in a moment. Moreover, as was the case with Andersen and Salomon Bros., a few dishonest individuals within an otherwise trustworthy organization can destroy public trust in the entire entity. We seek to develop technologies that reduce the cost of buliding a trustworthy commercial entity and eliminate the ability for dishonest insiders to abuse or otherwise endanger that trust, are efficient enough to be cost-effective, and simple enough to be adopted in real-world business settings.

We thus explore a tradeoff between theoretically complete security and practical efficiency, via what we call a *partially trusted third party* (PTTP). This PTTP is superior to an ordinary trusted third party, because it issues proofs that its actions were correct given the inputs – in every case. Thus the PTTP is never *trusted to* "do the right thing": it is *forced to*. Moreover, we seek to build protocols in which the PTTP is only entrusted with private information after it cannot gainfully exploit that information.

The actual partial trust placed in the PTTP depends on the protocol in which it is used. For example, in a sealed-bid auction, we might trust the PTTP not to

reveal private information *after* the auction is over, but prevent it from learning any private information until all bids have been submitted. Thus, the auctioneer cannot reveal a bid to any favored bidder; the proofs of correctness prevent the auctioneer from manipulating the already submitted bids.

Even this partial trust can be improved by the use of software, hardware and network security to build secure "black boxes" whose network connections are audited for correct behavior. *Trusted Computing* [142] provides a hardware and software framework for trustworthy computation in which specialized, secure hardware runs digitally signed operating systems and applications to achieve high confidence that the program is being run as specified. However, we point out out that in such circumstances, the proofs of correctness are still important because they ensure that no bugs in the signed software yield inadvertently incorrect results. Thus we can limit trust in these architectures only to prevent information leakage, not to ensure correct operation.

1.1 Related Work

Because this dissertation is comprised of several chapters that consider quite different problems, we consider existing work related to each chapter within the introduction to that chapter, and do not summarize it here. Rather, in this section we consider work related to our approach to provably correct, secrecy-preserving computation, and its relationship with an existing body of significant work in computer science and related fields.

One of the most influential papers in modern computer science is Goldwasser,

Micali and Wigderson's 1986 work introducing "zero-knowledge proofs" [70]. In this groundbreaking work,[1] the authors advanced the idea that parties can *prove* a particular fact correct, or that they possess private knowledge of some information, revealing anything more than the statement they are making—in particular, no private knowledge needs to be revealed. For example, if I know a three-coloring for a map, I can prove using these techniques that I know a three-coloring, without revealing to a verifier any valid three-coloring. The verifier learns only (typically with overwhelming probability) that the map is three-colorable.

These protocols are typically *interactive proof protocols*, in which a prover and verifier go through a back-and-forth set of challenges and responses that convince the verifier of the truth of the statement being proven. Importantly, to be assured of true zero-knowledge, the verifier must be able to construct a *simulation* of the entire communication stream without the aid of the prover. If the verifier could have constructed the entire communications without any assistance, then nothing was revealed.[2] Blum, Feldman and Micali later introduced "Non-Interactive Zero-Knowledge" [97], in which a reference string of "random data" simulates the choices of the verifier in an interactive process. This simplifies the process, and sometimes prevents certain attacks in which a dishonest verifier can learn something from the prover.

In this work, we employ "zero-knowledge" style techniques that allow computa-

[1] According to Rabin, before its publication the paper was rejected more than once by reviewers who could not understand why it would be useful to prove you knew something but not reveal what it was you knew.

[2] In extensions to zero-knowledge proofs beyond the scope of this work, relaxations to the requirement to construct a simulation without help, e.g. allowing up to k bits of "hints" to be transmitted, provide for more powerful protocols.

tions to be performed and proven correct on a collection of private values held by several different parties, for example, bidders in an auction or traders in a securities exchange, without revealing facts about the private values not implied by the result of the computation. Many of the cryptographic ideas we rely on, where a prover proves facts about a computation without revealing anything useful, find their origins in non-interactive zero-knowledge proofs.

The first solutions to solving this type of problem emerged even before the idea of "zero-knowledge", with Shamir's secret sharing scheme [140] and multi-party computation (MPC) that allowed the parties to distribute shares of their private inputs among a trusted group (often the parties themselves). This group would then jointly perform a computation so that a minimum number of the parties would be required to reconstruct the result – unless that number of members of the group cheated, only the result would be reconstructed. Later, "verifiable secret sharing" approaches provided for the provable correctness we desire, preventing a malicious member of the group from corrupting the output of the computation [61, 122]. Feigenbaum et al. advanced this paradigm further by providing for computations of approximate results in multi-party settings [60], which offers benefits in real-world situations we consider later, such as revealing facts about a portfolio's composition without permitting the recipient of the information to "back out" useful data from knowledge of the function and an exact output in its domain (see Section 9.3.2).

In such verifiable multi-party computation models, where we obtain both secrecy and proofs of correctness of the result, the parties with private data[3] typically col-

[3]In secret-sharing MPC protocols, the number of parties who can reconstruct the secret, and the number required for its reconstruction (or computation thereon) is determined *a priori*.

laborate in every step of computing the provably correct result (see for example the extensions of Gennaro et al. [67, 68]). In these models who boast they need no trusted third party, provable correctness is sometimes achieved by composing the function to be computed with an authentication code, yielding an authenticated result. They need no single trusted third party because trust is distributed. Canetti [42] introduced this notion of *universal composability*.

In other related theoretical work, Beaver et al. consider "instance-hiding"; an instance-hiding proof system for a function f "is a protocol in which a polynomial-time verifier interacts with one or more all-powerful provers and is convinced of the value of $f(x)$ but does not reveal the input x to the provers [22]. In our work, we are interested in a multi-party context of instance-hiding, where the provers prove a value of $f(x_1, \ldots, x_n)$ to various verifiers, and, ideally, nothing about any input x_i, each of which comes from a different original party, is revealed to the provers. Our work is also related to the field of private information retrieval (PIR), where a database provider answers queries about private data it stores on behalf of clients, but cannot learn anything about the data [87].

Finally, we mention the technique of obfuscated circuits, in which a circuit is obfuscated beyond recognition so that a third party can run the program on private data and not learn useful information. Recent work by Barak et al. examines what is and is not possible using these techniques, and reviews the previous work on obfuscated circuits related to secrecy-preserving computation [20].

Although many offer theoretically complete security, all of these protocols are cumbersome to implement in a business setting, and are burdensome for evaluating

computationally expensive functions – such as NP-hard problems or approximations thereof – over the private data. In addition, in many settings, a partially trusted third party who computes a single solution of a difficult problem yields a solution with many "results", each of which is to be privately transmitted to a distinct verifier. One example of this is where each bidder receives a proof of their personal outcome in a sealed-bid combinatorial auction. In a traditional MPC setting, separate computations would need to be carried out *for each result* if no party is to indeed learn nothing but what they need to know.[4] In the case of a large combinatorial auction with hundreds or even thousands of bidders and items, going through a costly MPC protocol to compute the outcome of the auction separately and provide each bidder with her personal outcome could make a practically intractable problem even theoretically impossible.

Bolstering our claim, we know of none of these high-security protocols which have been implemented in any large-scale real-world setting. While implementations have been published, e.g. the Fairplay system by Malkhi et al. [96] and the implementation of secure auctions based on multiparty integer computation by Bogetoft et al. described in Section 8.1 [26], the intractability of computations and the business process complexity inherent in implementing such systems for thousands of bidders are, in our view, a likely explanation for why important theoretical results that have been around for three decades are not yet in widespread business use.

[4]It is certainly possible that a large-scale multi-party computation could be optimized so that useful intermediate results could be reused in computing each output, but some separate computation would nonetheless need to be run for each distinct output.

1.2 Summary of the Presentation

We situate our work as seeking a compromise between a completely trusted third party and complete security: a "partially trusted third party". We describe such a party who receives commitments to private data, opens them at a designated time, performs various computations on the private data, reveals the results of the computations to designated recipients, and proves these results correct without revealing anything further about the private data. This partial trust is limited to disclosure – we do not place any trust in our partially trusted party with respect to correctness of the results: the proofs guarantee correctness. Moreover, we can eliminate even that limited trust by use of specialized hardware and specially designed networks that limit steganographic[5] revelation of information even after the computation is complete.

Of course, when some results are to be kept private, it is even more difficult to prevent steganographic encodings of private information in the private communication of an outcome to a particular verifier colluding privately with the third party. Lepinski et al. describe the problem of steganographic attacks in zero-knowledge contexts, and propose a "Fair Zero-Knowledge" protocol to limit these attacks [92]. We present our various contributions constructively. First, we develop the cryptographic ideas underlying secrecy-preserving, provably correct computation, through a general model of these computations, a construction of this model using Paillier's probabilistic and homomorphic cryptosystem [114], and the construction of a time-lapse cryptography service.

Next, we explore applications to auctions. After a brief introduction motivating

[5]Hiding information in the representation of other information, such as encoding private data in a public "random" value.

the need for secrecy-preserving, provably correct auction protocols, we construct two cryptographic auction protocols. The first is a simple cryptographic auction of one or more identical items with generalized Vickrey prices. The second is a novel cryptographic combinatorial auction mechanism that allows the auctioneer to solve the challenging optimization problems efficiently, yet prove the results correct to third parties. This mechanism includes a method of independent interest for proving a solution to a mixed-integer or linear program correct while revealing very little information.

Finally, we explore applications in securities trading. After a similar introduction motivating the need for secrecy-preserving, provably correct mechanisms for securities trading, we construct two cryptographic securities exchange protocols. The first is a cryptographic limit order book: a continuous double auction where traders submit encrypted buy and sell orders, and the market operator proves its actions are correct on the encrypted orders. The second is a cryptographic combinatorial securities exchange that permits traders to submit baskets of securities that clear with each other; the unfilled orders are then placed in a special remainder basket that third parties liquidate for a commission. Using an independently useful protocol, the market operator accepts representative portfolios from several third parties, and proves how their risk would change if they accepted the remainder basket. The commission, a predetermined function of this change in risk, is then verifiably computed; the third party with the lowest commission fills the orders. Thus, the third party receives a fair commission for its services but no party learns any information which can be exploited until after all trades are already committed to.

Chapter 2

Secrecy-Preserving Computations by a Partially Trusted Third Party

We define here a general model of secrecy-preserving computations by a partially trusted third party without tying it to a particular underlying cryptosystem. Various cryptographic building blocks can be used to create practical and secure implementations of our generalization; we describe two such constructions in detail in Chapter 3 and [128].

In our model, the partially trusted third party can compute a large number of results by performing a single expensive computation, and then prove each particular result correct using a much more efficient computation. Moreover, this separation of computation and verification allows the third party to employ the most efficient methods to compute the result, limiting the computationally costly operations over encrypted data to checking the proofs of the claimed results' correctness.

Another benefit of our model is that the "result" can be easily tuned to reveal

minimum information. The result might be that a particular payment for a good is $42, or which bid in an auction is the largest, but it might also be that the makeup of technology stocks in an investor's portfolio lies between 17 and 17.5 percent (see Section 9.3.6) or that a particular bidder met an activity rule of increasing her bids in each round of an auction (see Section 7.4.1). Indeed, the result can easily be any *fact* about the data — unlike in many efficient multi-party settings (such as polynomials over finite fields) in which the final result must be an element of the computation's domain. (That said, the theoretically general results described above can of course yield any computable result with proper encodings.)

This construction has an important relationship with computational complexity theory: many problems for which computing a solution is (believed to be) difficult have trivial proofs of correctness for any particular solution. Indeed, for any NP-complete problem, there always exists an statistical zero-knowledge argument that a prover knows a solution [111], if one-way functions exist. Similar zero-knowledge proofs can efficiently reveal particular facts about the solution without revealing the entire solution. By only employing the more burdensome verifiable computation for *verification* of results, we are free to use highly efficient hardware and software for the *computation* of those results.

In the particular applications we consider, many linear and integer programming problems that arise in combinatorial auctions are extremely difficult to solve, but proving a particular answer to be optimal is easier. In many cases, we can solve these problems over private data using advanced algorithms, and then prove the solution correct, requiring far fewer computational operations for the verifiers than

computing the allocation. In other cases, such as solving an integer program to run a combinatorial auction, an allocation can be shown to be *feasible* by demonstrating it satisfies particular constraints. Validating whether a particular result satisfies these constraints can be done extremely efficiently, and can be performed using the methods we describe using practical computing power.

This framework also supports specialized applications in which a particular party A with a private information state wishes to know the change its information state would undergo by learning information from B, without knowing anything else about B's information, or even learning what the updated information state would be. In one of our concrete examples, party A holds a particular portfolio of equities and wants to know how its risk profile would change if it "bought" a portfolio of equities from party B, without learning the particular equities in B's portfolio or revealing to B anything about either its own holdings or even the result of the computation. We offer concrete, efficient examples of such information exchanges in this general framework in Chapter 9.[1]

We begin with a formal definition of our model. There are a set of "parties" with private data, a partially trusted "Evaluator-Prover"[2], (EP) and zero or more independent "observers" who verify the outcome but do not supply any inputs to the computation. Each party has (without loss of generality) one private value, a public and encrypted form of that value that can be used in verification computations, and a cryptographic commitment to the encrypted value that is both binding and hiding from even the EP (see 2.1.3). We then define the security assumptions we make, and

[1] If greater theoretical security is necessary, one can always employ more secure multi-party computation for these protocols.

[2] This term is due to Rabin [128].

the (partial) trust we place in each party.

Next, we describe the nature of the data used in our computations and verifications, and the assumptions we make about underlying cryptographic protocols supporting our model.

Finally, we outline the computations and verifications using these data, in particular the specific operations we support with high efficiency, and how these operations provide for both practical implementations of important commercial protocols and theoretical computability of any computable function.

2.1 Secrecy-Preserving, Provably Correct Computation

2.1.1 General Assumptions

While this work is in the general realm of security, we do not wish to clutter the essential aspects of our protocols with too many details. To that end, we make several reasonable simplifying assumptions that do not affect our results.

Communications. We assume that there are secure private communications channels between all entities in the computation. We further assume a public channel where data may be posted publicly for all to see (often called a "bulletin board"). In practice, one way of achieving the secure private channels is to equip each entity with verifiable digital signature and public-key cryptographic keys – that is, build a public key infrastructure (PKI); we can model the public channel with a verifiable, distributed bulletin board where digitally signed posts are sent to multiple servers

who publish them on a computer network, and we assume that a majority of the servers remains active and uncorrupted at any time.

Cryptographic Complexity Assumptions. We assume the existence of one-way functions and the resulting implications. We require this assumption for the existence and security of cryptographic hash functions and cryptographic protocols we employ in particular implementations of our protocol. At the present time, we assume a random oracle model for cryptographic hash functions. That said, important cryptographic techniques exist that do not require a random oracle assumption; see the discussion in the Cramer-Shoup cryptosystem [49].

Honest but curious parties submit data to the computation. In general, we assume a party enters each computation without malice, as it expects the possibility of benefit from the results. To this end, each party will perform the protocol as prescribed. Since after the initial data are submitted, the parties have no further input, our model does not suffer from so-called *protocol completion incentive problems* (see Section 6.1.1, [31]). While we reserve it for future work, we believe it possible to extend our protocols to provide for a higher degree of security in which some of the parties may be malicious, inspired by advances in verifiable secret sharing and multi-party computation [61, 122, 67, 48].

2.1.2 Entities Involved in the Computation

In our model, a set of n *parties* $P_1, \ldots, P_n$ have private data values $x_1, \ldots, x_n$. (If a party has more than one private data value, then we model it as multiple parties without loss of generality.)

We assume a set of *verifiers* defined for each result of the computation which may include one or more of the parties. The set of verifiers for each result is defined at the beginning of the protocol. These may, but need not, include any or all of the parties providing data.

The *Evaluator-Prover* EP is a partially trusted third party responsible for accepting private data, computing one or more functions on the private data, revealing each result to the designated verifiers, and proving each result correct using the public, encrypted forms of the private data.

Assumptions about the Evaluator-Prover

Our protocols force the evaluator-prover to provide the following guarantees:

- The EP cannot see private data before they are needed, although the parties may be required to commit to their values earlier in the protocol. The EP cannot disclose data before it sees them.
- The EP can compute a correct result in every case using the data provided, or determine that no result exists.
- The EP can prove the reported result correct in every case.
- The EP cannot generate a valid proof of an incorrect result.
    - Depending on the underlying protocol, these guarantees may be qualified, for example, under realistic cryptographic assumptions or with a negligible probability of error.

We assume the following about the EP:

- The EP destroys all data not required for proving the computation correct at the conclusion of the protocol.
- The EP does not reveal anything to any party other than the results entitled to it as mandated by the protocol.

2.1.3 How the Computation Works

Initialization

In this exposition, we consider, without loss of generality, a setting with a single computation C with inputs x_i for each party P_i, where $i \in [1, n]$, and a set of two corresponding results z_1, z_2, each of which is revealed to its respective verifier $\mathcal{V}_1, \mathcal{V}_2$ and proven correct by the respective verification function V_1, V_2.[3] Each party P_i also creates a random data string σ_i; the EP creates a random string σ_{EP}. These strings are used later to tie the hands of the EP when issuing proofs.

To begin, the evaluator-prover (EP) publishes the computation C that is going to take place and a description of the domain of results. He also publishes a commitment to his random data $\mathbf{Com}_{EP}(\sigma_{EP})$. The EP requests that each party P_i prepare an input x_i, a to-be-published encryption of that input $E(x_i, r_i)$, a commitment to that encrypted input $\mathbf{Com}_i(E(x_i, r_i))$, and a commitment to its random data $\mathbf{Com}_i(\sigma_i)$. We recall that parties with multiple inputs are represented as multiple parties without loss of generality. The encryption function $E()$ is efficient and public

[3] In practice, the EP might compute many computations on a single set of inputs and issue any number of results, such as a private sealed-bid auction where he computes each winner's payment and proofs of those payments for each winner and a proof of an insufficient bid for non-winners. We use a model with two computations and results to illustrate the protocol with minimal notational overhead.

and corresponds to a decryption function $D()$ private to the EP; the details of encryption and decryption depend on the underlying cryptographic framework. Thus the parties supply $\mathbf{Com}_1(E(x_1)), \ldots, \mathbf{Com}_n(E(x_n)), \mathbf{Com}_1(\sigma_1), \ldots, \mathbf{Com}_n(\sigma_n)$ The EP publicly acknowledges receiving each commitment.

The EP requests that the verifiers $\mathcal{V}_1$ and $\mathcal{V}_2$ submit requests for results z_1 and z_2, and responds to these requests with verification functions V_1 and V_2, respectively. V_1 takes $n+2$ arguments: the n parties' encrypted inputs, the claimed result z_1, and proof information π_1. The function $V_1(\pi_1, z_1, E(x_1), \ldots, E(x_n))$ outputs a single bit which is true if and only if the result z_1 carried out is correct, that is, $C_1(x_1, \ldots, x_n) = z_1$. V_2 is analogous. The verification functions must not reveal any fact about any private data not implied by the result during the course of their evaluation.

Computation

The EP first announces that the inputs are now fixed and will accept no more inputs. In some protocols, it may be necessary to have a period of time that allows parties to contest non-inclusion of their data if they do not appear in the list of inputs published by the EP; we assume that appropriate measures will be taken to ensure that all parties' input commitments are accepted and posted.

When it is time for the EP to obtain the inputs, the commitments to the inputs and random strings σ_i are publicly unlocked, whether by the parties themselves or an alternative system, such as Time-Lapse Cryptography as described in Chapter 4 or escrow with another trusted party.[4] At this point the encrypted inputs $E(x_1), \ldots, E(x_n)$

[4]Such an alternative system is useful for enforcing *nonrepudiation* of commitments by the parties.

are public and visible to all parties, all verifiers, and the EP. Any necessary random data for the computation σ is now fixed as the exclusive or (XOR) of all random strings committed to before the protocol begins: $\sigma = \sigma_1 \oplus \sigma_2 \oplus \ldots \oplus \sigma_{EP}$. Provided any one of the components σ_i is truly random, the string σ will be truly random.

A fixed "random data" string is necessary to prevent the EP from initiating steganographic communications[5] with an adversary by manipulating the random help values used in the proofs. If all random data to be used in the computation and proofs are committed to before the inputs are known to the EP, then the EP cannot engage in such an attack. However, in our general framework, we make a simplifying assumption that the random string of data used in the computation can be made public, but remind the reader that in some cases many random values will be committed to *a priori* and some will never be revealed.[6]

The EP privately decrypts the inputs $x_1, \ldots, x_n$, and computes the provisional results $\langle \hat{z}_1, \hat{z}_2 \rangle = C(x_1, \ldots, x_n, \sigma)$. The EP then computes the proof data π_1, π_2 (which may well be a side effect of the computation C) and sends $(\hat{z}_1, \pi_1)$ to the verifier $\mathcal{V}_1$, and $(\hat{z}_2, \pi_2)$ to the verifier $\mathcal{V}_2$. Any random data used in constructing the result or proof must be derived from σ via a predetermined protocol.

[5]That is, hiding meaningful data in communications that appear to be for other purposes, for example, agreeing to set certain bits of a particular "random" help value to reveal the winning bid to another losing bidder in a sealed-bid auction.

[6]For example, in our yet unpublished research on extensions to the the RST cryptosystem [128], not all random data used in the computation can be publicly revealed without compromising the security of the protocol, but any random data must be *committed to* before the inputs are revealed to the EP.

Verification

$\mathcal{V}_1$ verifies the claimed result $\hat{z}_1$ by computing the verification function V_1 on the now available encrypted data, the proof π_1, the random data σ, and the result. He accepts $z_1 = \hat{z}_1$ if and only if $V_1(\hat{z}_1, \pi_1, \sigma, E(x_1), \ldots, E(x_n))$ outputs true. We require for any system that π_1 reveal nothing about the inputs and that nothing can be learned from this process other than the claimed result $\hat{z}_1$ and whether $\hat{z}_1$ equals the true result z_1.[7]

The verification functions are determined by the computation, but are not identical to the computation. They only prove that the result is correct, and that the result and proofs were constructed deterministically (given the randomness from σ). For example, C might be to order n input bids using an $O(n \log n)$ sorting algorithm and pick the winner. However, verifying whether the resulting ordering of n bids is correct requires only $O(n)$ comparisons; the verification function V_1 can be a program that checks whether a given order is correct or a particular bid is maximal. Thus the verifier need not sort the encrypted bids on his own; the provided ordering is sufficient to satisfy him.

This is one of many examples in which the verification can be much more efficient than the computation itself. Another benefit illustrated by this example is that the single expensive sorting operation yields an ordering of all bids, and that single result can be re-used in many verification processes to offer an efficient verification to each bidder.

The verifiers then check that the verification functions V_1 and V_2 will satisfy their

[7] In some cases in practice, we may relax this requirement so that the proof information reveals minimal information.

requirements. Each verifier can see that the verification function verifies a result of the announced computation is correct based on the underlying cryptographic protocol. Typically the verification function involves the verifier computing a sequence of operations over public encrypted values; this sequence of operations corresponds to a sequence of operations over the private unencrypted values which the verifier does not know, but because of this correspondence, the verifier believes the verification function to be correct. Where exact results are provided, the verifier can check that the encrypted result he computed uniquely corresponds to the unencrypted result and proof information supplied by the EP. Where approximate results are provided, the verifier can check an interval proof that the encrypted result he computed lies within the bounds stated by the approximate results. See Chapter 3 and [128] for specific examples of cryptosystems that achieve these ends.

2.1.4 Supported Mathematical Operations

Our general framework of evaluator-prover (EP) supports extremely efficient support for verifiable (in)equalities, addition and multiplication of integers, with a constant-factor increased cost for these operations over rational numbers. To model the broadest representation, allowing any underlying system, we speak of "verification" and not "computation" of the results, although in the instances we describe in this work, verifiers typically perform their verification by first performing a computation over the encrypted values.

Because the applications in which we are most interested are well-served by integer arithmetic, we focus on specific integer operations that specifically meet the needs of

the applications we describe.

We can extend these operations to rational numbers, which can be represented as pairs of integers, via inequalities and multiplications of integers; the increased cost is generally at most one multiplication per operation. This extension to rational arithmetic allows us to efficiently prove certain results about linear and integer programs as described in Chapter 7. Naturally, supporting modular addition and multiplication of integers provides for the evaluation of any theoretically computable function, but we recall that the other less efficient results mentioned earlier offer superior security when a theoretical result is desired; our purpose in extending these methods to the rationals is to offer a practical and efficient computational paradigm.

The following integer operations are supported. We observe that these operations can apply to both constants and encrypted values by replacing any encrypted value with a "null encryption" of a constant.[8] For the following exposition, we omit the random help values from our notation for clarity; the notation $E(x_i)$ is shorthand for $E(x_i, r_i)$.

- (In)equalities:

  $=$ Given $E(x_i)$ and $E(x_j)$, verify that $x_i = x_j$ in zero knowledge.

  We write $E(x_i) \equiv E(x_j)$.

  $\neq$ Given $E(x_i)$ and $E(x_j)$, verify that $x_i \neq x_j$ in zero knowledge.

  We write $E(x_i) \not\equiv E(x_j)$.

  $\geq$ Given $E(x_i)$ and $E(x_j)$, verify that $x_i \geq x_j$ in zero knowledge.

[8] An encryption that renders a value compatible with arithmetic under a cryptosystem without hiding that value.

We write $E(x_i) \trianglerighteq E(x_j)$.

- The $\leq$ operation follows from $\geq$; $<$ and $>$ follow from these, respectively, by adding 1 to one of the compared values. The symbols $\trianglelefteq$, $\triangleleft$, and $\triangleright$ are similarly employed to indicate inequalities between encrypted values' plaintexts.

- Arithmetic Operations:

  $+$ Addition: Given $E(x_i)$, $E(x_j)$, and $E(x_k)$, verify $E(x_k) \equiv E(x_i + x_j)$.

  $\times$ Multiplication: Given $E(x_i)$, $E(x_j)$, and $E(x_k)$, verify $E(x_k) \equiv E(x_i \times x_j)$.

  $-$ Additive Inverse: Given $E(x_i)$ and $E(x_k)$, verify $E(x_k) \equiv E(-x_i)$. (Alternatively, verify $E(x_k + x_i) = E(0)$.)

  $\div$ Multiplicative Inverse:[9] Given $E(x_i)$ and $E(x_k)$, verify $E(x_k) \equiv E(x_i^{-1})$. (Alternatively, verify $E(x_k \times x_i) = E(1)$.)

  $|x|$ Absolute Value: Given $E(x_i)$ and $E(x_k)$, verify that $E(x_k) \equiv E(|x_i|)$. One straightforward way to do this is to use multiplication to prove $E(x_i \times x_i) \equiv E(x_k \times x_k)$ and $E(x_k) \trianglerighteq 0$, using the above operations.

[9] In the systems we describe in this work, operations are taken modulo a large number. When the modulus is composite, as occurs for example when the homomorphic properties of the Paillier cryptosystem support an EP, a multiplicative inverse does not exist for all nonzero integers. We can still assume in this case that the multiplicative inverse is verifiable in practice, because finding a noninvertible value is of negligible probability. Happening upon such a value would allow factoring the modulus, and thus breaking the security of the underlying cryptosystem; doing so is as improbable as factoring the modulus by random guessing.

2.1.5 Extending These Operations to the Rationals

Given efficient implementations of the integer operations above, we can derive equivalent operations on the rational numbers that are a constant factor more expensive, due to most operations requiring one or two additional verifiable integer multiplications. While we admit that a series of computations on the rational numbers could be performed using only integer operations by multiplying through by a common denominator, this requires advance knowledge of the entire computation. While that might be appropriate in certain contexts, a general framework for computation over rational numbers has clear advantages. Chief among these is that the entire computation need not be known in advance.

We represent each unencrypted rational number as a pair of integers $\langle x, y \rangle$ and encrypt a rational number by encrypting the two integers separately: $\langle E(x), E(y) \rangle$. We typically employ the more familiar notation of $\frac{x}{y}$ for a rational number, and notate its encrypted form as either $E(\frac{x}{y})$ or $\frac{E(x)}{E(y)}$, which we consider semantically equivalent. We offer efficient options for proving equality and inequality so that the proofs can avoid costly multiplications wherever possible.

- (In)equalities:

  $=$ Given $\frac{E(x_i)}{E(y_i)}$ and $\frac{E(x_j)}{E(y_j)}$, either:

  - Verify $E(x_i) \equiv E(x_j)$ and $E(y_i) \equiv E(y_j)$ (fast)
  - Verify $E(x_i \times y_j) \equiv E(x_j \times y_i)$

  $\neq$ Given $\frac{E(x_i)}{E(y_i)}$ and $\frac{E(x_j)}{E(y_j)}$, either:

  - Verify $E(x_i) \not\equiv E(x_j)$ and $E(y_i) \equiv E(y_j)$

  - $*$ Verify $E(x_i) \equiv E(x_j)$ and $E(y_i) \not\equiv E(y_j)$
  - $*$ Verify $E(x_i \times y_j) \not\equiv E(x_j \times y_i)$

- $\geq$ Given $\frac{E(x_i)}{E(y_i)}$ and $\frac{E(x_j)}{E(y_j)}$, either:
  - $*$ Verify $E(x_i) \trianglerighteq E(x_j)$ and $E(y_i) \equiv E(y_j)$
  - $*$ Verify $E(y_i) \trianglelefteq E(y_j)$ and $E(x_i) \equiv E(x_j)$
  - $*$ Verify $E(x_i \times y_j) \trianglerighteq E(x_j \times y_i)$

- The other inequalities follow analagously, including strict inequality ($E(x_i \times y_j) \rhd E(x_j \times y_i)$ holds iff $\frac{E(x_i)}{E(y_i)} > \frac{E(x_j)}{E(y_j)}$.)

- Arithmetic Operations:

  - $+$ Efficient Addition: To prove $\frac{E(x_i)}{E(y_i)} + \frac{E(x_j)}{E(y_j)} \equiv \frac{E(x_k)}{E(y_k)}$,
    verify $E(x_k) \equiv E(x_i + x_j)$ and $E(y_i) \equiv E(y_j) \equiv E(y_k)$.
  - $+$ General Addition: To prove $\frac{E(x_i)}{E(y_i)} + \frac{E(x_j)}{E(y_j)} \equiv \frac{E(x_k)}{E(y_k)}$,
    verify $E(x_k) \equiv E(x_i \times y_j + x_j \times y_i)$ and $E(y_k) \equiv E(y_i \times y_j)$
  - $\times$ Multiplication: To prove $\frac{E(x_i)}{E(y_i)} \times \frac{E(x_j)}{E(y_j)} = \frac{E(x_k)}{E(y_k)}$,
    verify $E(x_k) \equiv E(x_i \times x_j)$ and $E(y_k) \equiv E(y_i \times y_j)$.
  - $-$ Additive Inverse: To prove $\frac{E(x_i)}{E(y_i)} \equiv -\frac{E(x_k)}{E(y_k)}$, either:
    - $*$ Verify $E(x_k) \equiv E(-x_i)$ and $E(y_i) \equiv E(y_k)$
    - $*$ Verify $E(y_k) \equiv E(-y_i)$ and $E(x_i) \equiv E(x_k)$
  - $\div$ Multiplicative Inverse: To prove $(\frac{E(x_i)}{E(y_i)})^{-1} \equiv \frac{E(x_k)}{E(y_k)}$, either:

* Verify $E(x_k) \equiv E(y_i)$ and $E(y_k) \equiv E(x_i)$
* Verify $E(x_i \times x_k) \equiv E(y_j \times y_k)$

2.1.6 Observations on verifiable computation over rational numbers.

We observe that in many applications, the denominators of rational numbers that are computed using the above methods can grow quite large, particularly after many multiplication operations. In practice, a system building proofs of correctness may need to periodically "reduce" rationals to their lowest integer denominator to prevent this. This can be easily done: to reduce $\frac{E(x_i)}{E(y_i)}$, the EP privately computes d, the greatest common divisor of x_i and y_i, publishes $\frac{E(x_i/d)}{E(y_i/d)}$, and proves that $\frac{E(x_i)}{E(y_i)} \equiv \frac{E(x_i/d)}{E(y_i/d)}$ via the cross product as above.[10] It is most critical that the numerator and denominator do not grow to be larger than any limitation imposed by the modulus of any finite field underlying the cryptosystem.

In some cases involving complicated computations, again those with many multiplications, it may not even be possible to reduce an encrypted rational value to a representation with small integer components because the greatest common divisor of the numerator and denominator is too small. In this case, the prover can mitigate problems of very large integers in the rational representation by rounding to a representation with smaller coefficients, then proving that the rounding "error" is within some acceptably small bound. The bound is encrypted, and then the difference between the rounded value and the actual value is proven to be smaller than that bound

[10] That is, $E(x_i \times (y_i/d)) \equiv E(y_i \times (x_i/d))$.

using their encryptions and the methods described above.

For example, if a computation resulted in the encrypted value $E(\frac{301}{900})$, and we allowed a tolerance of $\frac{1}{500}$, then we could reduce the result to $E(\frac{1}{3})$, then prove using the above methods that the error is within the allowed tolerance:

$$\begin{aligned} E(\frac{300}{900}) &\equiv E(\frac{1}{3}) \\ E(\frac{301}{900}) - E(\frac{300}{900}) &\trianglelefteq E(\frac{1}{500}) \\ E(\frac{1}{900}) &\trianglelefteq E(\frac{1}{500}) \\ E(1 \times 500) &\trianglelefteq E(1 \times 900) \end{aligned}$$

Clearly, the errors from every "reduce" operation can also be collected and summed at the end of the computation to verify that the total error is within some acceptably small bound.

2.1.7 Approximating real values with rationals; proving acceptable error

In the event our work is used in the context of a high-performance solver, the results obtained may be floating point values, not rational numbers. However, as we have not developed a cryptographic framework for working with arbitrary precision arithmetic,[11] these values must be converted to rationals in order for us to prove them correct.

[11] One could conceive of a similar approach for arbitrary floating-point arithmetic, where one might represent an encrypted floating-point value as a pair of encrypted values. For example, one might represent an encryption of 00123.45600 as "before" and "after the decimal point": $\langle E(123), E(45600)\rangle$. One could also conceive of a representation encrypting the significand and the exponent, for example, 123.456 would be $\langle E(123456), E(2)\rangle$. Developing an ideal representation and protocols for provably

We can easily approximate a decimal floating point value with high accuracy by taking its decimal representation as the numerator and a power of 10 as the denominator.[12] For example, the value $\pi \approx 3.14159265358979323846$ can be accurately approximated as $\frac{314159265358979323846}{100000000000000000000}$.

In the present work, we typically use our tools to prove a rational result satisfies a set of linear constraints. If we convert from a floating point value or real number (for example, from a commercial solver), the error in conversion may mean the rational result fails to satisfy the linear constraints.

To ameliorate this, we can apply the above ideas to prove the constraints are satisfied with acceptable error. To prove an existing constraint $E(a/b) \lhd E(c/d)$, we publish an encryption of a public error constant $E(x/y)$, then prove the constraint is satisfied within that error by proving that $E(ab) \lhd E(c/d + x/y) \equiv E(a/b) \lhd E((cy + xd)/dy)$, using the operations already been developed above. A similar approach provides for proving that an encrypted rational value satisfies any linear constraint within a particular error. If desired, the error could be multiplied instead of added to prove the constraint is satisfied within a certain factor of error, rather than the absolute magnitude of any error.

2.1.8 Integral rationals; comparing "fractionality"

In our calculations, it may be necessary to prove a rational number represented as above is integral. To prove $E(a/b)$ is integral, the prover first computes $E(c/1)$

correct secrecy preserving computations on floating point values is clearly a challenge beyond the scope of this work.

[12] The base is of course arbitrary, and a careful choice of base may reduce rounding errors or the requirement to reduce to smaller denominators.

and proves that $a = bc$. He then reveals that the denominator of $E(c/1)$ is in fact 1; provided c remains secret, no further information is revealed.

During the branch and bound proofs described in Chapter 7, we may need to prove which of two rational values is "more fractional". Given two encrypted rational values $E(x_1/y_1), E(x_2/y_2)$, we define this to mean the one whose fractional component is closer to 1/2. While a more efficient approach may be possible, we choose the following method for simplicity of exposition.

First, construct two encrypted values $E(I_1/1), E(x'_1/y_1)$, and prove that $E(x_1/y_1) = E(I_1/1 + x'_1/y_1)$, that $E(I_1/1)$ is integral, and that for the fractional portion x'_1/y_1, $x'_1 \geq 0$, $y_1 > 0$, and $x'_1 < y_1$. Nothing else is revealed about I_1, x_1, x'_1 or y_1. Do the same to obtain the analogous $E(x'_2/y_2)$. This yields two positive encrypted fractions $E(x'_1/y_1), E(x'_2/y_2)$.

To prove which value is "more fractional", we need to identify which is closer to 1/2. In lieu of developing a general absolute value function, we instead subtract 1/2 from each fraction, then square the result using the above multiplication operation. The smaller the result, the more fractional the value; a value of 0 indicates the fraction was exactly 1/2.

Let $E(z_1/w_1) \equiv E(x'_1/y_1 - 1/2) \times E(x'_1/y_1 - 1/2)$, with analogous $E(z_2/w_2^2)$.[13] Then the inequality $E(z_1/w_1) \trianglelefteq E(z_2/w_2)$ is true if and only if x_1/y_1 is at least as fractional as x_2/y_2. If it is necessary to prove when neither is more fractional—a tie—then we prove $E(z_1/w_1) \equiv E(z_2/w_2)$.

[13]We use z_1/w_1 for visual comfort: $z_1/w_1 = ((2x'_1 - y_1)^2/(2y_1)^2)$.

Chapter 3

Implementation of Secrecy-Preserving, Provably Correct Computation using Paillier Encryption

Paillier's encryption scheme [114] is ideal for supporting the general computational framework described in Chapter 2. In this section, we show how to derive the operations required by that framework using Paillier's function.

Paillier's is a *homomorphic encryption* system, in which the result of an operation applied to two ciphertexts is a valid encryption of an operation (possibly the same one) applied to their plaintexts.[1] In cryptography, a *plaintext* is the original form of

[1]More formally, in a homomorphic encryption scheme, there exist operations $\oplus$ and $\otimes$ such that given ciphertexts $C_1 = E(x_1)$ and $C_2 = E(x_2)$, $C_1 \otimes C_2 = E(x_1 \oplus x_2)$. Paillier's encryption scheme is homomorphic in that $E(x_1) \times E(x_2) = E(x_1 + x_2)$.

an input; a *ciphertext* is the encryption of a plaintext.

Homomorphic encryption schemes enable computation with encrypted values without revealing any new information about the values themselves or the results of the computation. Given a set of ciphertexts, anyone can calculate a new ciphertext that is a valid encryption of a function of the associated plaintexts. For example, multiplying two ciphertexts yields a new ciphertext that decrypts to the sum of their plaintexts. Paillier's system employs a public/secret key pair, N and ϕ respectively. The secret key N is the product of two large prime numbers p and q, and its size is determined by the security requirements of the application. The secret key ϕ is $(p-1)(q-1)$. A 1024-bit public encryption key is widely considered sufficient for security until 2010 [69].

Paillier encryption is also a *probabilistic* encryption scheme. In particular, encryptions are performed with a random "help value" r that is used to achieve *semantic security*: given two plaintexts and their encryptions, one cannot tell which ciphertext corresponds to which plaintext without being able to decrypt them. Semantic security is critical to preserve the secrecy of the inputs both during their initial encryption and during the verification process, where both inputs and the values in the test sets, whose plaintexts are well known, must still remain secret.

The security of this scheme is founded on the "Decisional Composite Residuosity Assumption" (DCRA) [114].[2] The DCRA implies that if the public key N is difficult to factor, then it is also difficult to tell whether a particular number x is a number of the form $x = r^N \pmod{N^2}$ for some r. This assumption is related to the widely

[2] A number $x = r^N \pmod{N^2}$ is known as an N^{th} *residue* mod N^2. Because N is a composite number—the product of two primes—x is called a composite residue.

accepted assumptions underlying the security of RSA[3] [131], ElGamal [58], and Rabin [127] encryption, and is believed to be of similar computational intractability.

The Paillier encryption of a message x will typically be denoted $E(x, r)$, where the public key N is implicit and the help value r is made explicit. In discussion below, the help value r will sometimes be omitted to simplify notation where it is implicit or irrelevant, for example, $c = E(x)$.

The fundamental homomorphic properties of Paillier encryption are simple, yet powerful. Given only the encryption $E(x_1)$ and either another encryption $E(x_2)$ or a public constant k, anyone can compute the encryptions $E(x_1 + x_2)$, $E(x_1 + k)$, and $E(x_1 \cdot k)$ *without learning anything about* x_1, x_2, *or the secret key* ϕ. Second, a prover $\mathcal{P}$ who knows the secret key ϕ can also prove a full set of *equality and inequality relations* for two encrypted values $E(x_1)$ and $E(x_2)$, e.g., $x_1 = x_2$, $x_1 > x_2$, etc., again, without revealing anything about x_1 or x_2. Moreover, a party who encrypted two values using the public key n, $E(x_1, r_1)$ and $E(x_2, r_2)$, can prove these same relationships using the help values r_1 and r_2, *even if the secret key* ϕ *is unknown.* It is also possible to compare encrypted inputs to constants in a similar way. We continue our use of the notation $E(x) \trianglelefteq E(y)$ to mean "$x \leq y$ can be proven using encrypted values $E(x)$ and $E(y)$" and the similar notation $\trianglerighteq$ ($\geq$), $\triangleleft$ ($<$), and $\triangleright$ ($>$).

[3]The RSA problem is the task of computing m given only x, N and e where $x = m^e \pmod N$ and $N = pq$ for large primes p, q. It is believed that the RSA problem is as difficult as factoring N, but this is unproven. Knowing the factorization of N allows one to efficiently solve the RSA problem.

3.1 Paillier Encryption

3.1.1 Public/Secret Keys

As above, Paillier encryption uses an encryption key $N = p \cdot q$, where p and q are large primes. The decryption key ϕ is derived from the factorization of N, $\phi = \varphi(N) = (p-1) \cdot (q-1)$. We recall that $\varphi(N)$ is Euler's totient function, the number of integers relatively prime to N. It is also required that N is relatively prime to ϕ.

3.1.2 Encryption

To encrypt a plaintext x, first compute a random value r from the range $[1, N-1]$ such that $\mathbf{gcd}(r, N) = 1$, then observe that $(1+N)^x \equiv (1+xN) \pmod{N^2}$ and encrypt as

$$E(x, r) = (1 + xN) \cdot r^N \pmod{N^2} \tag{3.1}$$

This is derived as follows:

$$\begin{aligned}
E(x,r) &= g^x \cdot r^N \pmod{N^2} \quad \text{(by [114])} \\
&\quad \text{set } g = (1+N), \text{ a generator of } \mathbb{Z}_N^* \\
&= (1+N)^x \cdot r^N && \pmod{N^2} \\
&= \left(\tbinom{x}{0} N^0 + \tbinom{x}{1} N^1 + \tbinom{x}{2} N^2 + \ldots\right) \cdot r^N && \pmod{N^2} \\
&= (1 + xN + \alpha N^2 + \ldots) \cdot r^N && \pmod{N^2} \\
&= (1 + xN) \cdot r^N && \pmod{N^2}
\end{aligned}$$

3.1.3 Decryption

To decrypt $C = E(x, r)$, given decryption key $\phi = (p-1)(q-1)$, observe that $r^{N \cdot \phi} \equiv 1 \pmod{N^2}$ by Euler's Totient Theorem, and

$$\begin{aligned} C^{\phi} &= (1+N)^{x \cdot \phi} r^{N \cdot \phi} \pmod{N^2} \\ &= \left(\tbinom{x \cdot \phi}{0} N^0 + \tbinom{x \cdot \phi}{1} N^1 + \tbinom{x \cdot \phi}{2} N^2 + \ldots\right) && \pmod{N^2} \\ &= 1 + x\phi N + \alpha N^2 + \ldots && \pmod{N^2} \\ &= 1 + x\phi N \pmod{N^2} \end{aligned}$$

implying

$$x = \frac{(C^{\phi} - 1)/\phi \mod N^2}{N} \tag{3.2}$$

While this method is clear, we did not use this method when obtaining our empirical results elsewhere in our work (Section 6.5). Instead, we used a more efficient algorithm involving precomputation and Chinese remaindering, as described in Paillier's Ph.D. thesis [113].

3.1.4 Decryption with random help value r

It is also possible for some $\mathcal{P}$ who knows the r used to encrypt $C = E(x, r)$ to show $\mathcal{V}$ that x is the unique decryption of C by revealing r. $\mathcal{P}$ may know r either by having encrypted all the values used to compute C or by computing it via the decryption key ϕ. To recover x, $\mathcal{V}$ computes

$$x = \frac{(C \cdot r^{-N} \mod N^2) - 1}{N} \tag{3.3}$$

$\mathcal{P}$ can also recover random help r from $C = E(x, r) = (1 + xN) \cdot r^N \pmod{N^2}$ by use of the secret decryption key ϕ as follows. (Note that our computations are

modulo N and not modulo N^2 because r was taken from $\mathbb{Z}_N^*$.)

$$\begin{aligned} r &= C^{N^{-1} \pmod{\phi}} \pmod{N} \\ &= (1+xN)^{N^{-1} \pmod{\phi}} \cdot r^{N \cdot N^{-1} \pmod{\phi}} \pmod{n} \\ &= 1 \cdot r^1 \pmod{N} \end{aligned} \tag{3.4}$$

3.1.5 Uniqueness of Encryptions

Paillier's encryption scheme involves a bijection from $(\mathbb{Z}_N \times \mathbb{Z}_N^*) \leftrightarrow \mathbb{Z}_{N^2}^*$ [114].[4] Thus any integer in $\mathbb{Z}_{N^2}^*$ represents a single valid encryption of an integer $x \in \mathbb{Z}_N$ with random help value $r \in \mathbb{Z}_N^*$. Consequently, if $C = E(x, r)$, $C \neq E(x', r')$ for any $x' \in \mathbb{Z}_N$ and $r' \neq r$. (This requires, as stated above, that $\mathsf{gcd}(N, \varphi(N)) = 1$.)

$\mathcal{P}$ can attempt to cheat by providing a different random help value r'. Using r' instead of r in (3.3) will yield a different but invalid "decryption" x'. $\mathcal{V}$ must therefore verify the provided value r' is consistent with the known encryption C. This can be done by re-encrypting the derived value x' as $C' = E(x', r')$ and rejecting r' unless $C' = C$.

3.2 Mathematical Operations on Encrypted Values

The following definitions apply to any values encrypted as above. These properties are due to the homomorphic properties of Paillier's encryption scheme [114]. In these

[4] $\mathbb{Z}_N$ is the set of integers $[0, N)$; $\mathbb{Z}_N^*$ is the subset of $\mathbb{Z}_N$ relatively prime to N.

definitions we refer to a prover $\mathcal{P}$ who has the decryption key or all random help values for encrypted data, and a verifier $\mathcal{V}$ who does not.

Addition. Addition of two encrypted values:

$$E(x) \cdot E(y) = E(x + y) \pmod{N^2}$$

Adding a constant k to an encrypted value x is easily done by encrypting k with the random help value 1 and multiplying the two encryptions.

$$E(x) \cdot (1 + kN) = E(x + k) \pmod{N^2}$$

Multiplication by a constant.

$$(E(x))^k = E(x \cdot k) \pmod{N^2}$$

Negation. Implied by multiplication by a constant.

$$(E(x))^{-1} = E(-x) \pmod{N^2}$$

Comparison to a constant k. $\mathcal{P}$ can prove any encryption $C = E(k, r)$ is an encryption of k by revealing the help value r used to encrypt C. $\mathcal{V}$ then verifies that $(1 + Nk)r^N = C \pmod{N^2}$, because

$$E(k, r) = (1 + N)^k \cdot r^N \pmod{N^2} \tag{3.5}$$

This is of particular interest when $k = 0$. We remark that no encryption of a value other than zero is an N^{th} residue[5] mod N^2.

[5]To say that x is an N^{th} *residue* (mod m) means that there exists some value g such that $x = g^N \pmod{m}$. See also Footnote 2.

3.2.1 Secrecy-Preserving Equality and Inequality Proofs

Equality comparison. Given two ciphertexts $C_1 = E(x_1, r_1)$ and $C_2 = E(x_2, r_2)$, $\mathcal{P}$ can prove $x_1 = x_2$ without revealing any additional information.Both $\mathcal{P}$ and $\mathcal{V}$ compute $C' = C_1 \cdot C_2^{-1} \pmod{n^2} = E(x_1 - x_2, r_1/r_2) = E(0, r_1/r_2)$. $\mathcal{P}$ then proves C' is an encryption of zero as above by revealing r_1/r_2.

Inequality comparison. Given two ciphertexts $C_x = E(x)$ and $C_y = E(y)$, $\mathcal{P}$ can show $x > y$ and $x \geq y$. Because our values x and y are integers mod N^2, we can prove $x > y$ by showing $x \geq y + 1$, provided $y \neq N - 1$. Due to the homomorphic properties of Paillier encryption, $E(x+1) = E(x) \cdot (N+1) \pmod{N^2}$, and so adding 1 to a value in its encrypted form is trivial. Thus, all ordering comparisons can be reduced to the ability to prove $x \geq y$. We first specify that x and y must be in the range $[0, 2^t)$ for $2^t < N/2$. This can be proven as described in Section 3.2.3. Then, to prove $x \geq y$, both $\mathcal{P}$ and $\mathcal{V}$ calculate $E(x - y) = E(x) \cdot E(y)^{-1} \pmod{N^2}$, and $\mathcal{P}$ proves $0 \leq (x - y) < 2^t < N/2$ from $E(x - y)$. If in fact $x < y$, then $(x - y)$ will wrap around mod N^2 so that $(x - y) \geq N/2$ and no such proof is possible. This principle is also detailed in Section 3.2.3.

To show that $x \neq y$ given $E(x)$ and $E(y)$, without revealing anything about their relative magnitude, there are a few possible solutions. One simple solution (for $|x - y| < \sqrt{N/2}$) using our other primitives is to verifiably compute $E(z) \equiv E(x - y)$, then prove that $z^2 > 0$ by showing that $E(z \cdot z) \rhd E(0)$.

3.2.2 Secrecy-Preserving Proof of Products of Encrypted Values

Because Paillier encryption does not enable the secrecy-preserving multiplication of two encrypted values as it does addition, we require a method that allows a prover $\mathcal{P}$ with three plaintexts u, v, and w such that $uv = w \pmod{N}$ to prove this fact to a verifier $\mathcal{V}$ who has Paillier encryptions $E(u)$, $E(v)$, and $E(w)$, respectively. Dåmgard et al. [51] propose another solution to this problem; our solution is in the spirit of our other cryptographic primitives.

A *Multiplication Test Set* (MTS) for $E(u, r)$, $E(v, s)$, and $E(w, t)$ is a set of 8 elements:

$$\{E(u_1, r_1), E(u_2, r_2), E(v_1, s_1), E(v_2, s_2),$$
$$E(w_{i,j}) = E(u_i v_j, p_{i,j}) \mid i, j \in \{1, 2\}\}$$

where $u = u_1 + u_2 \pmod{N}$ and $v = v_1 + v_2 \pmod{N}$.

In each MTS, u_1 and v_1 are chosen uniformly at random from $\mathbb{Z}_n$; u_2 and v_2 are correspondingly defined, as above, so that $u = u_1 + u_2 \pmod{N}$ and likewise for v.

Clearly, if given encryptions as in MTS and

$$w_{1,1} + w_{1,2} + w_{2,1} + w_{2,2} = w \pmod{N} \tag{3.6}$$

then in fact $uv = w \pmod{N}$. But for $\mathcal{P}$ to prove and for $\mathcal{V}$ to verify all the relationships included in the MTS, $\mathcal{P}$ must reveal u_1, u_2, v_1, and v_2, which would consequently reveal u and v. Thus we adopt for an interactive proof the following challenge and partial revelation proof. $\mathcal{P}$ constructs and sends MTS. $\mathcal{V}$ randomly

chooses a challenge pair (i, j), say, $(1, 2)$, and sends it to $\mathcal{P}$. In this case, $\mathcal{P}$ reveals r_1, s_2, and $p_{1,2}$. This allows $\mathcal{V}$ to decrypt $E(u_1)$, $E(v_2)$, and $E(w_{1,2})$, and directly verify that $u_1 \cdot v_2 \equiv w_{1,2} \pmod{N}$. $\mathcal{P}$ further reveals:

$$R = r_1 \cdot r_2 \cdot r^{-1} \pmod{n}$$

$$S = s_1 \cdot s_2 \cdot s^{-1} \pmod{n}$$

$$p = p_{1,1} \cdot p_{1,2} \cdot p_{2,1} \cdot p_{2,2} \cdot t^{-1} \pmod{N}$$

$\mathcal{V}$ by use of R verifies $E(u_1) \cdot E(u_2) \cdot E(u)^{-1} \pmod{N^2} = E(0, R)$, i.e., verifies $u = u_1 + u_2 \pmod{N}$ and similarly $v = v_1 + v_2 \pmod{N}$ via S. Finally, $\mathcal{V}$ verifies $E(w_{1,1}) \cdot E(w_{1,2}) \cdot E(w_{2,1}) \cdot E(w_{2,2}) \cdot t^{-1} \pmod{n^2} = E(0, p)$, thereby verifying that (3.6) holds.

If MTS was not proper then the probability of $\mathcal{V}$ uncovering this by the random choice of (i, j) is at least $\frac{1}{4}$. Thus the probability of $\mathcal{P}$ meeting the challenge when $uv \neq w \pmod{N}$ is at most $\frac{3}{4}$. This implies that if m MTS's are used and $\mathcal{P}$ meets all m random challenges then the probability of $\mathcal{P}$ cheating is smaller than $(\frac{3}{4})^m$. In practice, the prover will verify multiplications by repeating these zero-knowledge proofs until the desired probability of error is achieved.

3.2.3 Verifiable, Secrecy Preserving Interval Proofs

The method we describe here is simple to describe, but not as efficient as other methods advanced in other work. We refer the reader to Kiayias and Yung [81] for a discussion of more efficient interval proofs using a method first described by Boudot [30]; Damgård and Jurik [51, 79] also discuss Paillier interval proofs, and Lipmaa et al. [94] present similar solutions for efficient interval proofs in auctions.

Rabin et al. [128] also employ a somewhat different and more efficient technique inspired by Brickell et al. [36] for interval proofs in the RST scheme.

In order to prove that $a \geq b$ for two values a and b, we can show that $a, b < N/2$ and then that $(a - b) \pmod{N} < N/2$ as described above.[6] This works because if a and b are less than $N/2$ and a is greater than b, then clearly $a - b < N/2$; if a is *less* than b, then $a - b$ will "wrap around" modulo N and must be a large number, that is, $a < b \Rightarrow a - b \pmod{N} > N/2$.

Thus, with a single additional primitive to prove that $x < N/2$ given only an encryption of x, we can prove inequalities of values using only their encrypted forms. Given ciphertext $C = E(x, r)$ we want to prove that $x < 2^t$ for some t such that $2^t < N/2$. That is, we we want to be able to verify that a value x is smaller than some agreed upon bound 2^t, without revealing any information about x. The value of t determines the number of bits of resolution available to parties in selecting their inputs.

We perform the test as follows:

A valid *test set* TS for the assertion "$C = E(x, r)$ is an encryption of a number $x < 2^t < N/2$" is a set of $2t$ encryptions:

$$TS = \{G_1 = E(u_1, s_1), \ldots, G_{2t} = E(u_{2t}, s_{2t})\} \tag{3.7}$$

where each of the powers of 2: $1, 2, \ldots, 2^{t-1}$ appears among the u_i exactly once and the remaining t values u_j are all 0. Each test set's elements are randomly ordered.

By use of a test set TS, the prover $\mathcal{P}$ can prove that $x < 2^t < N/2$ as follows:

[6] Because our mathematical operations are over the integers modulo a large number, a small negative number is the same as a large positive number, and vice versa. For example, $13 \equiv -2 \pmod{15}$.

Range Protocol. Let $x = 2^{t_1} + \ldots + 2^{t_\ell}$ be the representation of x, a sum of distinct powers of 2. $\mathcal{P}$ selects from TS the encryptions $G_{j_1}, \ldots, G_{j_\ell}$ of $2^{t_1}, \ldots, 2^{t_\ell}$, and further $t - \ell$ encryptions $G_{j_{\ell+1}}, \ldots, G_{j_t}$ of 0. Note that:

$$\left(E(x,r)^{-1} \cdot G_{j_1} \cdot \ldots \cdot G_{j_t}\right) \pmod{N^2} = E(0,s) \tag{3.8}$$

is an encryption of 0 with help value $s = (r^{-1} \cdot s_{j_1} \cdot \ldots \cdot s_{j_t}) \pmod{N}$ if and only if indeed $x = 2^{t_1} + \ldots + 2^{t_\ell}$ and the G_{j_h} were chosen as stated. Now since $\mathcal{P}$ has the decryption key ϕ and thus knows the help value r, then he can hand over to $\mathcal{V}$ the set $\{G_{j_1}, \ldots, G_{j_t}\}$ and the above help value s. $\mathcal{V}$ can now verify on her own that (3.8) holds and deduce that $x < 2^t < N/2$.

The above protocol reveals nothing to $\mathcal{V}$ beyond $x < 2^t < N/2$, because TS is a random set, in actual implementation a randomly permuted array of the elements in question. Consequently $\mathcal{V}$ has no information about *which* encryptions of powers of 2 are included in $\{G_{j_1}, \ldots, G_{j_t}\}$ Furthermore, the inclusions of $t - \ell$ encryptions of 0 hides even the number of non-zero bits in the binary representation of x. Finally, the random factors $s_{j_1}, \ldots, s_{j_t}$ present in the test set's encryptions combine to a uniformly random s, which completely masks any information about the help value r in the encryption $E(x, r)$. Consequently no information about x is revealed.

There is, however, a problem with the above protocol in that $\mathcal{V}$ does not know that $\mathcal{P}$ has presented her with a true test set. This is overcome as follows. For ease of understanding, we first describe an interactive verification protocol, then modify it for non-interactive use. The idea is to use a "cut and choose" procedure in which the prover commits to a number of test sets and allows the verifier to choose and inspect multiple test sets and make sure that they are each valid. Finally, the remaining test

sets are all used to complete the proof. An early, possibly the first, use of this idea was presented by Rabin [126].

Tamper Proof Interactive Verification of $x < 2^t < N/2$. First, the prover $\mathcal{P}$ creates $2v$, say for $v = 20$, test sets $TS_1, \ldots, TS_{2v}$, and presents those to $\mathcal{V}$ claiming that they are all valid. Verifier $\mathcal{V}$ randomly selects v test sets $TS_{i_1}, \ldots, TS_{i_v}$ and requests that $\mathcal{P}$ reveal all the encryptions by revealing all the corresponding help values. $\mathcal{V}$ verifies all the encryptions and checks that every TS_{i_h} is valid. If any verification fails, the process is aborted. Otherwise, there now remain v unexamined test sets, call them $TS_{j_1}, \ldots, TS_{j_v}$. $\mathcal{P}$ now completes v repetitions of the above **Range Protocol**, and establishes that $x < 2^t < N/2$ by use of each of the above remaining v test sets. If all verifications succeed then $\mathcal{V}$ accepts that indeed $x < 2^t < N/2$.

The only way that $\mathcal{P}$ can cheat is if all the above remaining v test sets are invalid, which requires that initially the $2v$ test sets comprised v proper test sets and v improper ones and, furthermore, when examining the test sets, $\mathcal{V}$ randomly chose all the v proper ones. The probability of such an unlucky choice is $\binom{2v}{v}^{-1}$. In our example of $v = 20$, that probability is, by Sterling's Theorem, about $\sqrt{\frac{20\pi}{2^{40}}} < \frac{8}{10^{12}}$. Thus, we have a zero-knowledge protocol for $\mathcal{V}$ to verify interactively with $\mathcal{P}$ that $x < 2^t < N/2$, when given a ciphertext $E(x, r)$ such that the inequality actually holds.

Tamper Proof Non-Interactive Verification of $x < 2^t < N/2$. We prefer to adopt the following non-interactive method[7] to establish the validity of test sets in our scheme. Suppose that there are (as in Section 6.3.2) $2k$ range-of-values tests to

[7]Non-interactive zero knowledge was introduced in [97].

perform. Once the inputs to the computation are fixed, $\mathcal{P}$ publishes $4kv$ test sets. (For expository convenience, we proceed below with our assumption of $v = 20$.)

When the computation is set up, the parties providing inputs and the EP are also asked to commit to a random string, which will be revealed after the inputs are fixed and after the EP commits to test sets. We recall from Section 2.1.3 that each party providing input and the EP submit random strings that are combined into a random data source σ.

The $80k$ test sets posted on the Bulletin Board are then segmented into $2k$ groups of 40 test sets each, i.e., the first 40 test sets, the next 40 test sets, etc. The random bit-string σ is then used, in combination with a fixed rule available to all participants and posted at the start of the computation to the bulletin board, to select 20 test sets from each group. This random selection replaces the random selection by the verifier $\mathcal{V}$ employed in the interactive proof and allows the proof to work without interaction.

Bulk Verification of Test Sets

Because in practice a computation will require large numbers of test sets, we may accelerate the non-interactive verification process by verifying all the test sets to be used for a computation *en masse*, which requires a smaller percentage of the test sets be revealed and thereby made unusable.

We have already shown how the EP can use a test set to prove both that for any encrypted inputs $E(x_1)$ and $E(x_2)$, $\{x_1, x_2\} \leq 2^t$ and $x_1 > x_2$, provided $2^t < N/2$. However, the verifier $\mathcal{V}$ needs to know that the test set the EP uses to prove this is correctly constructed in order to believe the proof.

In a traditional zero-knowledge proof (ZKP) setting, the EP would present $\mathcal{V}$ with several test sets in a "cut-and-choose" protocol, and $\mathcal{V}$ would then select at $\mathcal{V}$'s own discretion some of the test sets for the EP to reveal. In our setting, it is impractical for the EP to perform real-time ZKP's of input correctness to all of the verifiers. Therefore, we employ a technique where instead of the verifier choosing the test sets to reveal, we derive randomness from the test sets themselves and use that randomness to define both which test sets will be revealed, and the order in which other test sets will be used to verify the computation. This means that the EP can *publish* a ZKP of the correctness of the test sets that anyone can verify. This can even be done asynchronously, i.e., the test sets used to prove a computation correct can be verified correct before the inputs are fixed.

All of the test sets must be of identical form for a computation on inputs of maximum size 2^t. Each test set will contain t encryptions of powers of 2: $2^0, \ldots, 2^{t-1}$, and t encryptions of 0. For visual comfort, we will use examples where $t = 32$, accommodating inputs in a range of over 4 billion values. Because any input or comparison of inputs can be verified using such a test set, we will prepare a single very large collection of test sets that will be used for all comparisons in a computation.

We demonstrate with very high probability that for collections of sufficient size, after revealing 20% of the collection, no more than 10% of the remaining unrevealed test sets are improper. Assuming we draw from the remaining test sets uniformly at random, the probability of a correctness proof of s succeeding, i.e., all s sets are improper is $< 10^{-s}$.

If we select and reveal 500 test sets uniformly at random in a collection of 2500,

the probability that all 500 will be correct and 200 (or more) of the remaining 2000 are incorrect is $< 7 \times 10^{-19}$. We can then prove an input or comparison between inputs with probability of error $< 10^{-10}$ by drawing 10 of the remaining 2000 test sets uniformly at random and proving correctness on each of them.

We can achieve a reasonable "random" ordering from the test sets using the random data string σ; let R be a predefined substring of σ of suitable length for this purpose.

Step 1. The EP privately creates 2500 test sets $TS_i, i \in [0, 2499]$, each of which is comprised of encryptions of 64 small values, $\{c_{i0}, \ldots, c_{i63}\} = \{E(0) \times 32, E(2^0), \ldots, E(2^{31})\}$. The EP creates a secret random permutation $\pi_i(0 \ldots 63) \in \{0 \ldots 63\}$ for each TS_i for each of the encrypted values in the test set and privately stores the plaintexts, random help values r and exponentiations thereof $r^N \pmod{N^2}$.

Step 2. The EP creates a permutation $\rho(0 \ldots 2499) \in \{0 \ldots 2499\}$ of an ordering of the 2500 test sets using the random data in R according to the protocol published at the beginning of the computation.

Step 4. The EP reveals the first 500 test sets defined by the ordering ρ. Verifiers will be given a reasonable specified time (depending on the size and complexity of the computation) to verify the correctness of these test sets. If a test set is discovered to be invalid, the EP creates 2500 new test sets and the protocol is begun anew at Step 1.

Step 5. If all 500 test sets are correct, then ρ (excluding the revealed test sets) defines the random ordering of the unrevealed test sets that are used to prove the correctness of the inputs and computations.

3.3 Securely Computing a Product of Random Help Values

In our cryptographic combinatorial clock proxy auction in Chapter 7, we use the following method for bidders to jointly compute the product of their random help values used to encrypt their demands; this allows them to obtain the random help value that will decrypt the encrypted aggregate demand. We recall that the encrypted aggregate demand was created as the product of all bidders' encrypted demands. This protocol assumes nothing other than secure communications channels among the bidders.

Before continuing, we remark that in the general case it may be viewed as superior to have parties other than the parties providing the inputs compute the aggregate values to prevent malicious parties from intentionally disrupting the protocol. This might take the form of bidders secret-sharing their random help values with another group of parties (such as those conducting a time-lapse cryptography service). We also refer the reader to Damgård and Jurik's related threshold protocol for Paillier encryption [51], invented for the similar task of computing sums of ballots in electronic voting.

In the following protocol, the bidders first construct random shares of their random help values (their inputs), distribute these shares among all the bidders, construct intermediate factors from these shares, then multiply the factors together to that yield the aggregate product without revealing anything about the initial inputs.

Formally, at the end of round t, for each good G_j, each bidder B_i breaks her

random help value for her encrypted demand for that good $E(s_{ij}^t, r_{ij}^t)$ into n shares $r_{ij1}^t, \ldots, r_{ijn}^t$ whose product equals $r_{ij}^t (\pmod N)$. The remaining operations in this section are also assumed to be performed $(\pmod N)$.

Each bidder chooses nonzero shares $r_{ij2}^t, \ldots, r_{ijn}^t$ at random from $\mathbb{Z}_N$, and verifies that each value is relatively prime to N. This is because in practice, the group $\mathbb{Z}_N^\times$ (the positive integers less than N relatively prime to N) is not known to the bidder: knowing the membership of that group is equivalent to factoring N—thus compromising the security of the cryptosystem. This poses no practical problem because a bidder happening upon a value that is *not* relatively prime to N is as improbable as the bidder factoring N by random guessing. This is a safe assumption since we assume the cryptosystem is secure.

She then computes the final share r_{ij1}^t such that

$$r_{ij1}^t \equiv \frac{r_{ij}^t}{\prod_{k=2}^n r_{ijk}^t} (\pmod N),$$

that is, so the product of all her shares equals her input. She then sends a vector of shares of her encrypted demand for each good $\langle r_{i1k}^t, \ldots, r_{imk}^t \rangle$ to each bidder B_k (including herself).

Now each bidder B_i has the following set of vectors: $\langle r_{11i}^t, \ldots, r_{1mi}^t \rangle, \langle r_{n1i}^t, \ldots, r_{nmi}^t \rangle$ She computes another vector whose elements are intermediate factors: the product of the shares provided by the other bidders and m of her own: $\langle \hat{r}_{i1}^t = \prod_{k=1}^n r_{k1i}^t, \ldots, \hat{r}_{im}^t = \prod_{k=1}^n r_{kmi}^t \rangle$. She then publishes $\langle \hat{r}_{i1}^t, \ldots, \hat{r}_{im}^t \rangle$. These factors are indistinguishable from a collection of random values.

Once all bidders have published these intermediate factors, every bidder computes and publishes the vector containing the product of each of these factors for every

good: $\langle \hat{r}_1^t = \prod_{i=1}^n \hat{r}_{i1}^t, \ldots, \hat{r}_m^t = \prod_{i=1}^n \hat{r}_{im}^t \rangle$ By the associative law of multiplication, this vector now holds the product of all bidders' random help values for each good: $\hat{r}_j^t = \prod_{i=1}^n r_{i1}^t$ (note $r_{i1}^t \neq \hat{r}_{i1}^t$.) These values can decrypt the sum of all bidders' demands—the aggregate demand—without revealing anything about any particular bidder's demand.

Chapter 4

Time-Lapse Cryptography

The notion of "sending a secret message to the future" has been around for over a decade. Despite this, no practical solution to this problem is in common use. We name, construct and specify an implementation for a cryptographic primitive, "Time-Lapse Cryptography", with which a sender can encrypt a message so that it is guaranteed to be revealed at an exact moment in the future, even if this revelation turns out to be undesirable to the sender. Our solution combines new ideas with Pedersen distributed key generation, Feldman verifiable threshold secret sharing, and ElGamal encryption, all of which rest upon the single, broadly accepted Decisional Diffie-Hellman assumption. We develop a Time-Lapse Cryptography Service ("the Service") based on a network of parties who jointly perform the service. The protocol is practical and secure: at a given time T the Service publishes a public key so that anyone can use it, even anonymously. Senders encrypt their messages with this public key whose private key is not known to anyone – not even a trusted third party – until a predefined and specific future time $T+\delta$, at which point the private key is constructed

and published. At or after that time, anyone can decrypt the ciphertext using this private key. The Service is envisioned as a public utility publishing a continuous stream of encryption keys and subsequent corresponding time-lapse decryption keys. We complement our theoretical foundation with descriptions of specific attacks and defenses, and describe important applications of our service in sealed bid auctions, insider stock sales, clinical trials, and electronic voting.

4.1 Introduction

First proposed by Timothy May [99], many attractive protocols have been proposed to encrypt messages to send into the future, usually under a name like "timed-release cryptography". We coin the phrase "Time-Lapse Cryptography" to distinguish protocols like ours, in which a fixed amount of time elapses between the ability to send a message (encrypt) and retrieve it (decrypt), from other methods in which only estimates of or lower bounds on elapsed time can be given.

4.1.1 Setting and Objectives

The setting for our service is as follows: At time T, Alice wishes to send Bob a message m so that Bob may decrypt it only at or after a specified future time $(T+\delta)$. This decryption will be possible without any further action by Alice.

Our "Time-Lapse Cryptography Service" ("the Service") makes this possible. At or before time T, the Service publishes a public key PK along with a statement that its corresponding private key SK will be revealed at time $T+\delta$. Alice uses PK to encrypt m with random help r using a probabilistic encryption scheme and sends

the ciphertext $c = E_{PK}(m, r)$ to Bob. She is now committed to the content of the message, although Bob cannot yet see it. At time $(T + \delta)$, the Service reconstructs and publishes SK, which Bob obtains and uses to decrypt c and recover m. (Of course, Alice, if she so wishes, can always reveal m early by sending Bob m and r.)

The primary objectives of our Service are as follows:

- The Service publishes a public key PK associated with a start time T, duration δ. It includes authenticating information with which users can unequivocally determine the authenticity of PK, T, and δ.
- The private key SK corresponding to PK remains completely secret until time $T + \delta$.
- At time[1] $T + \delta$ the Service publishes the decryption key SK, along with authenticating information that allows any user to unequivocally determine the authenticity of SK.
- The Service is resistant to attacks that attempt to generate insecure public keys, prevent the generation of public keys, reconstruct the private keys early, or prevent accurate and timely reconstruction of private keys.

4.1.2 Summary of Contributions

We offer a complete description of a service and associated protocols that enable Time-Lapse Cryptography as described in Section 4.1.1. The Service we describe is

[1]Plus a negligible delay ϵ for reconstruction described in Section 4.3.1.

simple and easy to understand by anyone with an elementary cryptography background.

It is anonymous: the Service knows nothing about who might be using it; this increases privacy and eliminates any incentive for early private key reconstruction if the Service were to know a key were used for an important purpose.

The Service allows the originator of a message complete control over when the recipient may decrypt it, while guaranteeing that the recipient may decrypt the message at a specific future time.

The protocols rely only on well-studied and widely accepted cryptographic primitives: Pedersen distributed key generation (DKG) [122], Feldman verifiable secret sharing (VSS) [61], and the ElGamal cryptosystem [58].[2] Conveniently, the security of all three of these primitives rests on the widely believed assumption of the hardness of the Decisional Diffie-Hellman problem. This offers an elegant consistency and simplicity to the security of our proposal.

Our protocols guard against such attacks as: the Service being able to prematurely reveal the decryption key; the Service refusing to reconstruct the decryption key at the required time; users of the Service getting inconsistent views of the stream of public and private keys. We detail these and other attacks in Section 4.2.2. It will be clear from our construction that all these attacks are rendered impossible under generally accepted assumptions.

Our work also names and describes this protocol as a new cryptographic primitive that may be useful in complex protocols. This primitive can be viewed as a

[2]As described later, we recommend the use of more recent variants of these DKG and VSS protocols to eliminate specific attacks which may slightly bias the uniform distribution of the public keys.

simple cryptographic commitment that is concealing *and* that cannot be repudiated. Alice is not only bound to not change content of the message; unlike in some other commitment schemes, such as those based on cryptographic hash functions, Alice furthermore may not prevent the message from being read by refusing to reveal the message (input to the hash function). When a binding commitment is required, Alice's digital signature on the ciphertext of a time-lapse encrypted message yields a commitment binding Alice to the still-secret content of the message.

An additional contribution of our work is a detailed enough description that will serve as a basis for an implementation of a time-lapse cryptography service, including details of and defenses against real-world attacks. We plan to deploy such an implementation in the coming months.

4.1.3 Extension to Paillier Keys

A useful extension to the present work is the use of distributed key generation schemes for other cryptographic keys, most notably composites of two large primes ("RSA keys"[3]) used in the Paillier homomorphic encryption scheme described in Chapter 3. Boneh and Franklin first proposed an efficient distributed RSA key generation protocol [27]; Frankel et al. offer a more robust formulation in [63].

While we plan to extend our protocol to RSA/Paillier keys in future work, the existence of these secure DKG protocols support a protocol in which TLC keys can be used directly to encrypt inputs in our computation model. That said, the ElGamal scheme we describe can be used even in a computation model employing Paillier-

[3]RSA encryption is the best-known scheme to assume the hardness of factoring such composites for its security.

encrypted values by first encrypting the values with the Evaluator-Prover's public Paillier key, then encrypting those encryptions yet again using a time-lapse protocol.

4.1.4 Applications

While we do not attempt to anticipate all of the possible applications that may be discovered for such a service, many useful applications already motivate its creation. We remark that time-lapse cryptography is not appropriate or sufficient for some applications. Time-lapse crypto is not appropriate when the sender wishes to revoke a message—indeed, nonrepudiation is an important property of our system. Furthermore, our protocol makes no guarantees about the correctness, authenticity, or suitability of the encryption of a particular message. Other protocols, such as interactive zero-knowledge proofs, may complement time-lapse cryptography where such requirements exist.

Bids in sealed-bid auctions. As mentioned in the preface, our original motivation for this work came from our earliest joint work with David Parkes and Stuart Shieber on cryptographic auctions [115]. In that auction protocol, we realized the need for bidders to issue commitments to their bids that were secret to even the auctioneer during the auction but could not be repudiated after the close of the auction. This prevents a type of abuse in which the auctioneer decrypts some bids and instructs favored bidders to refuse to unlock their bids (for example, because they offered far too much.)

Using our service, a bidder doubly encrypts her bids, first with the auctioneer's public key PK_{AU} and then the public key PK_S published by the time-lapse encryption

service S. This creates the ciphertext $c = E_{PK_S}(E_{PK_{AU}}(\mathbf{Bid}))$, which is digitally signed by the bidder and published on a bulletin board. Thus no one, including the auctioneer, knows anything about her bid until either she reveals the random help value she used in $E_{PK_S}()$ or the appropriate amount of time elapses. No action of any bidder can prevent the auctioneer from decrypting her bid or the public from using her encrypted bid $E_{PK_{AU}}(\mathbf{Bid}))$ in verification protocols after the time-lapse expires.

We make extensive use of time-lapse cryptography in the sealed-bid and continuous double auction protocols throughout this dissertation.

Insider stock trades. An insider to a publicly-traded company could be legally obligated to issue advance commitments to stock transactions to mitigate the potential for abuse of inside information, as well as to protect the insider from false accusations of misuse of inside information. In certain circumstances, it is desirable that those commitments stay secret until shortly after the execution of the transaction in question. Clearly, a commitment that does not guarantee nonrepudiation does not suffice since an insider may publish in advance a concealed commitment to a trade and then refuse to reveal it in the event the trade is no longer desirable to him. If an insider encrypts his transaction in advance using a time-lapse cryptography service, he can always be legally compelled to complete the transaction although the details of the transaction remain secret until the appointed time. We suggest a protocol in which insiders issue their advance directives daily (say, for various lengths of time in advance) using the Service. These directives may be to buy, sell, or do nothing, which are indistinguishable under the semantic security of ElGamal. In this way an insider reveals no information to the market; while it is intuitive that disclosure of this in-

formation could hurt the insider, Fishman claims that insiders can exploit disclosure rules due to the fact that "the market cannot observe whether an insider is trading on private information or for personal portfolio reasons [62]." If, in fact, insiders always trade for liquidity reasons and never (illegally) trade on private information, then disclosure of their trades *should* have no market impact (trades accompanied by credible claims they are for liquidity reasons are known as "sunshine trades"; see see Admati et al. [7]). Current SEC regulations require *ex post* disclosure for certain insiders, in part due to the argument that advance disclosure reveals too much information. The time-lapse cryptography Service answers this argument.

Data collected in clinical trials. In order to preserve the integrity of clinical trials, the data collected during such a trial may be encrypted using a time-lapse cryptography service. Because many of these trials are funded by companies who stand to make or lose significant amounts of money depending on their outcome, there is the potential for pressure to achieve a positive result. Use of our Service can mitigate this bias without revealing confidential information about the study before it is complete. Time-lapse cryptography prevents unethical scientists from cheating, and benefits ethical scientists by protecting them against false claims of fraud or pressure from their funders to achieve a particular outcome. Our protocol's property of early revelation also enables data collected in such trials to be revealed early in the case of necessity, for example, in cases that a drug is so effective it would be immoral not to offer it to the control group.

In one setting, scientists' data collection process uses our Service to encrypt data directly as they are being collected, for example, by diagnostic devices or computer

user interfaces. The scientists would not be able to see the data collected until the conclusion of a phase of the study; this prevents observations of trends in early data collection from affecting future data collection practices.

In another setting, clinical data would be provided to the scientists in raw form immediately and to an auditing board encrypted via time-lapse cryptography. The scientists would preserve the confidentiality of their data during the study to prevent leaking of information by the auditing board, but would know that any tampering with results would be discovered after the expiration of the time-lapse.

Electronic Voting. In some voting applications, the publication of intermediate results may be undesirable, as it could unduly influence other voters or election officials. If votes are encrypted using time-lapse cryptography during an election, results can be kept completely confidential until polls close, as well as being assuredly revealed promptly when required.

4.1.5 Related Work

Solutions that do not have a fixed decryption time generally involve expensive sequential computations ("time-lock puzzles")[4] to recover an initial message, ensuring that the recipient cannot recover the data before some length of time, such as those proposed by Rivest et al. [132]. Other solutions that do not guarantee fixed time release are made possible by partial key escrow, first described by Bellare and Goldwasser in [23].

A number of ideas inspiring our approach use known encryption techniques in

[4]Merkle [104] is generally credited with inventing these "puzzles".

which the decryption key is kept secret until a fixed revelation time. Blake and Chan [25] describe the "Timed Release Encryption Problem" as a sender encrypting a message such that only a particular receiver can decrypt that message, and that only after a specific release time has passed, as verified by a single trusted, third-party time server. They solve this problem with a bilinear pairing on a Gap Diffie-Hellman group, which requires reasonable cryptographic assumptions.

Blake and Chan's solution is similar to those employed in identity-based cryptography. Other work sharing this connection is the work by Cheon et al. [46] formalizing "secure timed-release public key encryption" and its equivalence to strongly key-insulated public key encryption. Their solution, also based on a bilinear map, requires a trusted "timed-release public server" that periodically publishes information, based on a private secret, that enables decryption of previously encrypted texts. Dodis and Yum [55] proposed a related protocol in which digital signatures become verifiable only at a fixed future time t upon publication by a trusted third party of "some trapdoor information associated with the time t."

Our solution is also related to "token-controlled" public key encryption, first introduced by Baek et al. in [17]. In token-controlled encryption, messages are encrypted with both a public encryption key and a secret token, and can only be decrypted with the private decryption key after the token is released. Time-lapse cryptography and token-controlled encryption share many important applications, and in fact an approach similar to time-lapse crypto could be used as a means of securely generating and distributing the secret tokens with distributed trust.

Rivest et al. [132], in addition to time-lock puzzles, offer the first description

we know of the use of a secret decryption key for time-lapse cryptography; in their scheme, a trusted third party creates and distributes public and private keys at appropriate times. Our protocol is similar, but replaces their trusted third party with a network of parties and an assumption that no fewer than a specified number of these parties need to be trusted.

Di Crescenzo et al. [50] employ a trusted time server and a new primitive called "conditional oblivious transfer" to send messages into the future where the server never learns the sender's identity. (It does learn the receiver's identity.)

4.2 Preliminaries and Assumptions

Our service consists of the following major components:

- A network of n participating parties $P_1, \ldots, P_n$
- Distributed key generation of the public and private keys
- Verifiable threshold secret sharing of the private key
- Secure multi-party reconstruction of components of the private key
- Reconstruction and publication of the private key
- Secure public and private bulletin boards for posting of intermediate and final results

The protocol is conducted by a "Time-Lapse Cryptography Service" ("the Service") consisting of n parties $P_1, \ldots, P_n$. The protocol allows for the possibility that

these parties may only be intermittently available. It also allows for the existence of adversaries that may attempt to disrupt the protocol in various ways. Call the generation of a public key and the corresponding reconstruction of the private decryption key an "action" of the Service. We assume a threshold t such that during any one action, at most $t-1$ parties may disrupt the protocol by revealing secret information, submitting false information, or refusing to participate in the action. Any such party will be informally referred to as being *improper*. We also assume that during the entire action, at least t parties strictly follow the protocol. Such parties will be informally referred to as being *proper*. This implies that $n \geq 2t-1$.

We postulate that for the ElGamal encryption, there is a publicly agreed-upon cyclic group G and generator $g \in G$ of prime order q. For this work we assume that $2q+1$ is a prime p, and that G is the set of quadratic residues modulo p; hence, all elements of G other than $\{1,-1\}$ have order q. This ensures semantic security vis-à-vis quadratic residuosity.

We further assume that p and q are selected with appropriate attention to cryptanalysis, so that the encryption scheme used is resistant to known attacks involving vulnerabilities of particular "unsafe primes." Without loss of generality, we will refer to only one group G and public generator g for both ElGamal encryption and the verification of shared secrets. Other groups G are possible, most notably elliptic curve groups that offer improved efficiency.

4.2.1 Implementation Considerations

The Service will be implemented on a network of autonomous computers, each of which represents a party P_i in our protocol. Each party follows the protocol described below; it obtains the schedule of public key generation and private key reconstruction from a set of manager computers we next describe.

For further efficiency, reliability and resistance to attacks, we employ a small network M of K managers that act as a "managing team" for the Service. The role of the managing team is to create the schedule of the public and corresponding private keys to be produced by the Service; to maintain an internal bulletin board for use by the parties comprising the Service; and to maintain a public bulletin board for users of the Service. Integrity of these bulletin boards is achieved by each manager maintaining his own copies of these two bulletin boards. Parties and users will look at messages posted on each of the managers' copies of the bulletin boards and determine the correct values by a majority of postings.

The authoritative time for all actions shall come from an assumed universally accessible clock (Section 4.2.4), and no party or manager shall rely on an internal clock. All computers comprising the Service should be maintained by administrators with experience in security considerations and running operating systems with up-to-date security patches.

4.2.2 Resistance to Attacks

Up to $t - 1$ improper parties P_i may attack the Service in various ways. We describe in detail our protocol's resistance to the following attacks by these improper

parties at the appropriate phase of our protocol in Section 4.4.

- Sabotaging the joint construction of a valid, random PK
- Posting an incorrect value of PK
- Prematurely reconstructing SK (prior to time $T + \delta$)
- Sabotaging the reconstruction of SK at time $T + \delta$

In addition, an improper party can attack the distributed key generation algorithm we describe by introducing a slight bias into the distribution of possible public keys. We refer the reader to work by Gennaro et al. [68] for a complete description of this attack and a modified algorithm that prevents it. Those implementing the protocol may wish to periodically review cryptology research on distributed key generation in order to guard against new attacks.

We also point out that improper parties or users of the Service may mount denial of service attacks by attempting to overload the Service with internal or public bulletin board postings or requests for keys. The managers of the Service can prevent such attacks by appropriate rationing of postings and requests. Of course, there exist other known possible denial of service attacks, and corresponding countermeasures, that are outside the scope of this work.

4.2.3 Security Assumptions

The protocol employs the ElGamal encryption scheme [58]. ElGamal's scheme *is* semantically secure under chosen plaintext attacks (CPA): adversaries can encrypt as many messages as they want and gain no information about the private key or any

other encrypted message. ElGamal is known to be trivially malleable and hence insecure under chosen ciphertext attacks (CCA-1). We do not view this as a security risk, because no ciphertexts can be decrypted with the private key before its reconstruction and publication, and it is expected at that time that all ciphertexts encrypted with that key can be decoded by anyone. Malleability is not of concern in our protocol, because it can be avoided by signing encrypted messages via an appropriate, nonmalleable digital signature scheme.

We assume that each party P_i uses a computer that accurately and secretly performs the computations we describe and securely stores all P_i's secret data. We assume the parties back up data in some secure way for disaster recovery, though it must be a method that makes stealing the secrets from backups at least as difficult as compromising the hosts themselves.

4.2.4 Communications Assumptions

We assume that each party P_i can communicate privately and secretly with any other party P_j. For example, each party may have a public/private cryptographic key pair and all parties will know every other party's public key.

In addition, our protocol will require posting of various intermediate steps and results. The parties will employ the internal bulletin board provided by the managing team for that purpose, as described in Section 4.2.1. Posting of any message m by a party P_i will always be accompanied by P_i's digital signature $SIGN_i(m)$.

We also assume a universally accessible and tamper-resistant clock, such as provided by the US NIST, that determines times for actions taken by the Service.

4.2.5 Summary of ElGamal Encryption

As described above, we assume a publicly known group G and generator thereof g. The Service creates and publishes an ElGamal public key $PK = g^x$ as described later; the private key is $SK = x$.

To encrypt a message m, Alice first obtains the public key $PK = g^x$ and creates a random help value $y \xleftarrow{R} [1, q-1]$. She then computes the ciphertext c as a pair: $c = (g^y \pmod p, m \cdot g^{xy} \pmod p)$.

Alice then sends this pair c to Bob. By elementary algebra, Bob can recover m when the Service publishes the private key x or Alice later sends him the random help value y.

4.3 How the Service Works

For a less formal introduction to our protocol, we recall the reader to our high-level overview in Section 4.1.1.

4.3.1 What the Service Does

The Service creates, publishes and maintains "time-lapse cryptographic key structures" that represent public time-lapse cryptography keys with a specific lifetime. The Service may generate these structures on a periodic basis for public convenience; for example, each day it might release keys with a lifetime of 1 week, or every 30 minutes release keys with a lifetime of 2 hours. These schedules are posted by the managers to the public bulletin board. In addition, the Service can accept requests from clients

to generate new keys with a particular lifetime; the managers accept these requests and post them on the public bulletin board. The parties P_i construct the key structures according to the protocol, individually sign them, and publish the signed key structures on the public bulletin board.

For each key required by convention or client request, the Service will generate a key structure $K_{ID} = (ID, T_{ID}, \delta_{ID}, PK_{ID})$ consisting of a unique identifier ID, a publication time T_{ID}, a "time-lapse" δ_{ID}, and a public key PK_{ID}. Each party P_i publishes the key structure and signature thereof $(K_{ID}, SIGN_i(K_{ID}))$ on the public bulletin board.

At time $(T_{ID} + \delta_{ID})$ the Service will reconstruct and publish the associated private key SK_{ID}. The public key and private key for K_{ID} are related by the equation $PK_{ID} \equiv g^{SK_{ID}} \pmod{p}$. It is assumed that the generator g is public. It is crucial that the private key SK_{ID} is known to no one, and never reconstructed, before the appropriate time. As before, each party P_i publishes the reconstructed private key and signature thereof $(SK_{ID}, SIGN_i(SK_{ID}))$ on the public bulletin board.

There is a subtle issue in that reconstruction of the private key is not in fact instantaneous. In practice, the Service will begin reconstruction of the private key SK_{ID} at time $(T_{ID} + \delta_{ID})$ and publish SK_{ID} at time $(T_{ID} + \delta_{ID} + \epsilon)$ where ϵ is the time required to reconstruct the private key. We assume that ϵ can be made negligible in comparison to any time-lapse δ_{ID} and will be on the order of a fraction of a second, and therefore we generally assume $\epsilon = 0$ for convenience. At the beginning of the time lapse, we assume that the time T_{ID} is an upper bound on the time when the public key is released, and that the Service may release a key required at time T_{ID} at

any time at or before T_{ID}.

4.3.2 What the Clients Do

When Alice wishes to send Bob a message m, she requests or selects an appropriate key structure K_{ID} from the Service. Alice does not need to identify herself in any way in order to do this; because the Service publishes the key structures on the public bulletin board, Alice may use any mechanism for obtaining the public key structure, e.g. a friend or an anonymous Web proxy server. Alice then verifies the published digital signatures $SIGN_i(K_{ID})$ match the published key structure K_{ID} for a minimum of a threshold t parties P_i, and that these parties' K_{ID} are identical. This guarantees that PK_{ID} is the public key generated by all the proper parties, and its corresponding decryption key SK_{ID} will be subsequently reconstructed and correctly posted by all the proper parties.

To send the message, Alice encrypts m using ElGamal encryption; she creates a random help value $y \xleftarrow{R} [1, q-1]$ and privately sends Bob the pair $c = (g^y \pmod p, m \cdot PK_{ID}^y \pmod p)$ as well as the index ID of the key structure K_{ID} whose public key she used. Alice may at this stage apply other appropriate cryptographic primitives, such as a digital signature or message authentication code, depending on the application. If Alice wishes to send a longer message than can be accommodated by the group G, she may use our protocol to encrypt and send a secret key for a block cipher and encrypt her actual message with that block cipher, or she may break her message up into smaller chunks and encrypt each one.

Alice now has no ability to stop Bob from decrypting her message. Bob receives c

and stores it, then waits for Alice to send y or for time $(T_{ID} + \delta_{ID})$, whichever comes first. If Alice sends him y, he decrypts m using $g^{PK_{ID}}$ and y; if she does not, he obtains PK_{ID} from the Service and decrypts m using that.

4.4 Protocol for the Parties P_i in the Service

We use a standard distributed key generation (DKG) algorithm as described by Pedersen [122], and employ Paul Feldman's simple verifiable secret sharing (VSS) scheme [61] to guarantee the authenticity of the generated keys.[5]

Throughout this section we shall designate a set of "qualified" parties Q which are the parties that have complied completely with the protocol and not been disqualified for any reason. It will turn out that for any action (i.e. the construction of an encryption key PK and the subsequent reconstruction of the corresponding decryption key SK), Q will include all proper parties. Consequently, $|Q| \geq t$ at all times.

4.4.1 Distributed key generation

Whenever a fixed "preparation interval" before a posted key generation time T is reached, each party P_i begins the protocol. For example, the Service might schedule a 1-week key to be released each day at 10:00 am Eastern Time; the parties begin preparing this key a few minutes ahead of schedule so that it can be released at or before 10 am. It will be seen later on that parties to the Service may be disqualified during the creation phase of the public key by demonstrably violating the protocol.

[5] See Section 4.2.2 for a brief discussion of a subtle attack and a reference to a modified algorithm defending against it.

We shall refer to the set of parties that were *not* disqualified as the set Q of "qualified parties". It will turn out that all proper parties (and possibly some improper parties) P_i will be members of Q, and that every proper party will have the same view of (value for) Q.

Each party P_i should choose a random $x_i \stackrel{R}{\leftarrow} [1, q-1]$. This x_i constitutes P_i's candidate *component* of the private key. It will turn out that the private key will be $x = \sum_{i \in Q} x_i \pmod{q}$. Each P_i should then compute $h_i = g^{x_i} \pmod{p}$ and post $(h_i, SIGN_i(h_i))$ on the internal bulletin board. It will turn out that the public key will be $h = \prod_{i \in Q} h_i \pmod{p}$. This h_i is P_i's candidate component of the public key. Any party P_i who does not post h_i is disqualified. Obviously, all proper parties will have the same view of which parties were disqualified at this point.

4.4.2 Sharing the private key components

In order to ensure that the private key x corresponding to the public key h will be correctly reconstructed at time $T + \delta$, we have to protect against the possibility that improper parties will refuse to reveal their component x_i of the private key x or reveal a false value instead of x_i. This is achieved by use of verifiable threshold secret sharing. During this phase, further parties P_i may be disqualified.

Each party P_i should create a random polynomial of degree $k = t - 1$ in $F_q[z]$:

$$f_i(z) = x_i + a_{1i}z + a_{2i}z^2 + \ldots + a_{ki}z^k$$

The secret key component is $f_i(0) = x_i$. Each party P_i should compute secret shares $x_{ij} = f(j)$ and verification commitments $c_0 = h_i = g^{x_i}, c_1 = g^{a_{1i}}, \ldots, c_k = g^{a_{ki}}$. (All commitments c_i are computed $\pmod{p}$.) Each P_i then privately sends to all

$P_j, j \in [1,n]$, $(j, x_{ij}, SIGN_i(j, x_{ij}))$ and posts on the internal bulletin board signed commitments $(c_0, SIGN_i(c_0)), \ldots, (c_k, SIGN_i(c_k))$. Every P_j can now verify that x_{ij} is a correct share by checking $(*)$:

$$(*) \quad g^{x_{ij}} \equiv c_0 c_1^j c_2^{j^2} \ldots c_k^{j^k} \pmod{p}$$

(In our proposal, index j is the argument to the polynomial for all P_j.)

At this point an improper P_i can disrupt the process in one of two ways. First, he may send P_j an incorrect share x_{ij} of his component x_i. In this case, P_j posts the triple $(j, x_{ij}, SIGN_i(j, x_{ij}))$ on the internal bulletin board. The proper parties will check whether x_{ij} is valid according to $(*)$. If it is an invalid share, then P_i is disqualified. All parties can check whether x_{ij} is a valid share according to $(*)$. All proper parties will arrive at the same conclusion as to whether P_i should be disqualified.

Second, P_i may have failed to send P_j the share x_{ij}. In this case P_j posts a signed protest to the internal bulletin board. P_i is then required to reveal x_{ij} on the internal bulletin board by posting a signed message $(j, x_{ij}, SIGN_i(j, x_{ij}))$. Every party can then verify the posted share x_{ij} according to $(*)$. If it is invalid, then P_i is disqualified. Again, all proper parties will reach the same conclusion as to the disqualification of P_i.

Putting all the above together, it is clear that all proper parties now have the same view of the value Q, the set of qualified parties.

Despite the above posting of some shares, the secrecy of the private key is preserved until time $T + \delta$. Consider first shares x_{ij} of the private key component x_i of a proper party P_i. Only improper parties P_j will (unjustly) demand revelation of such shares. Thus, just a total of at most $t - 1$ shares of x_i will be posted. By the properties

of Shamir secret sharing [140], the component x_i remains random to the improper parties, and any observer of the internal bulletin board. Of course, the improper parties can always circulate the shares they received anyway: an adversary gains nothing by this revelation. Next, consider shares x_{ij} of an improper party P_i who refuses to send P_j its share. The posting of P_i's shares may reveal x_i. However, even if every improper P_i would broadcast its component x_i of the private key x, the private key remains secret until the components x_j of the proper parties are revealed and this happens only at time $T + \delta$.

4.4.3 Publishing the public key

Now, each qualified party P_j holds the public key h, a component x_j of the private key x, and shares x_{ij} for all qualified parties P_i. These latter shares are kept for the reconstruction of any missing components x_i that are unavailable at the conclusion of the protocol if P_i is unavailable or corrupted.

Every qualified party $P_j, j \in Q$ forms $h = \prod_{i \in Q} g^{x_i} \pmod{p}$ and the key structure $K_{ID} = (ID, PK_{ID} = h, T_{ID}, \delta_{ID})$. and posts $(K_{ID}, SIGN_j(K_{ID}))$ on the internal *and* public bulletin boards. A simple analysis shows that all the parties proper during this action will post the same value for K_{ID}. The number of such proper parties strictly exceeds $n/2$. Consequently, any user viewing the public bulletin board can unambiguously extract a valid value for K_{ID}. The public key PK_{ID} can now be used for time-lapse encryption. Clearly, users can and should verify the digital signatures on data posted on the public bulletin board.

4.4.4 Reconstructing and publishing the private key

At the appointed time ($T_{ID} + \delta_{ID}$) for the reconstruction of the private key SK_{ID}, by definition, all parties proper for this action will correctly participate. Consequently, at least t proper parties will do so. Parties consult the public bulletin board maintained by the managers to obtain the list of reconstruction times, and begin the reconstruction protocol when the time $T_{ID} + \delta_{ID}$ for reconstructing SK_{ID} is reached on the universal clock.

First, every party P_i should publish its component x_i of the private key $x = SK_{ID}$ to the internal bulletin board. By definition, all proper parties do so. Note that even after this is done, certain components x_i previously provided by some $P_i \in Q$ may be missing if the party P_i in question is in fact improper. Every proper party then checks that for every $P_i \in Q$, the posted x_i satisfies the equation $g^{x_i} \equiv h_i \pmod{p}$, where h_i is as published in the previous step. For any $P_i \in Q$ who has not posted x_i or for whom this verification fails, the parties need to reconstruct the correct x_i. Obviously, by definition, at least the parties proper within this action will do so. Note that the parties $P_i \notin Q$ are of no interest since their candidate shares did not enter into the construction of the private key x.

Now, every party P_j should post the x_{ij} it received from P_i during the distributed key generation phase described in Section 4.4.2.

Note that at this point, every proper party P_j has either received a verified x_{ij} from P_i which it posts, or in the "Sharing the Private Key" phase (Section 4.4.2) of the protocol, demanded of P_i to post to the internal bulletin board the share x_{ij}. Furthermore, that posted share was verified. This holds because otherwise P_i would

have been disqualified and not included in Q.

In summary, every proper P_j now sees on the internal bulletin board at least t valid shares x_{ij} of P_i's component x_i of the private key $x = SK_{ID}$. The party P_j uses any t shares x_{ij} to reconstruct x_i by polynomial interpolation.

After this is done, every proper party P_j has all the components x_i for all the parties $P_i \in Q$. Every such P_j now computes the sum $SK_{ID} = x = \sum_i x_i \pmod q$ and publishes $(ID, SK_{ID}, SIGN_j(ID, SK_{ID}))$ to the public bulletin board. Now, there will be strictly more than $n/2$ signed postings agreeing on the value of SK_{ID}. Consequently, any user looking up the value of SK_{ID} can unequivocally determine it, even if improper parties attempt to sabotage the reconstruction or the posting of the private decryption key.

The *Handbook of Applied Cryptography* [102] offers a concise description of polynomial interpolation in Section 12.71 in its description of Shamir's (t, n) threshold secret sharing scheme [140].

4.4.5 Proactive renewal of components and shares

Because there may be applications where a time-lapse cryptographic key has a very long life (for example, a year or more), it may be prudent to periodically redistribute the shares of each party's component of the private key and shares thereof for additional security. With such a system in place, an adversary has a limited time to obtain the required number of secret components before the components are renewed and past components are no longer useful. A protocol for doing so for ElGamal cryptosystems is described by Herzberg et al. [74] and related work. This enhancement

can be directly combined with the protocols we advance in this chapter.

4.5 Conclusions and Future Work

We have developed a simple, practical and clear protocol that solves the problem of "sending a secret message into the future" and a Service that implements it. Our formal treatment firmly establishes this idea as a useful cryptographic primitive; previous work and our suggested applications demonstrate broad applicability. Our work goes beyond a purely theoretical foundation and describes how our Service might be implemented in practice with important practical details, including resistance to specific attacks. We have isolated the fundamental elements of the "Time-Lapse Cryptography" primitive in our construction. This allows for established primitives to perform additional cryptographic functions. For example, the sender Alice of a time-lapse encrypted secret to Bob can restrict subsequent revelation solely to Bob by further encrypting the ciphertext again with Bob's public key; she can achieve non-malleability via a message authentication code; she can apply her digital signature to prove she sent the message, etc. Thus we have a clean, independent primitive that is easy to understand and employ in more complex protocols.

Plans for future work include a complete implementation of our Service on a distributed network of computers made available for public use. During this process we will improve the details of our specification and deepen our understanding of the practical security of the underlying protocols we employ.

We also anticipate that we and others will invent and describe novel applications of this technology once it is publicly available. For example, the homomorphic properties

of ElGamal and Paillier encryption may be useful in a time-lapse setting. We have also considered modified time-lapse cryptography protocols in which we retain the properties of sender anonymity and guaranteed future decryption if the sender does nothing, yet allow the sender to delay decryption until a later time upon request to the Service. One application of this extension is to the encryption of a will, in which the testator wishes to postpone its revelation until required.

Chapter 5

Introduction to Cryptographic Auctions

In recent years, auctions and electronic marketplaces have been used to facilitate trillions of dollars in trade in the world economy [59]. Auctions, in particular, are often adopted to promote the ideal of competitive pricing and economic efficiency [100, 29]. Previously used for rare goods, or for time-sensitive goods (e.g., flowers and fish), auctions can now be harnessed for all kinds of commercial transactions [105]. Auctions see especially wide use for the procurement of goods and services by firms and governments [54, 76, 151]. We also note that more and more auctions of all kinds are electronic, and operate over the Internet, which reduces the cost of participation and enables worldwide competition.

Individual procurement events in the private sector, for instance, the procurement of truckload services by Procter and Gamble, approach US $1 billion in transaction value. To give a sense of the scale of procurement in the public sector, Asker and

Cantillon [13] estimate public procurement in the European Union at about 16% of its GDP; by this estimate public procurement comprised $2 trillion of trade in 2006 [152]. Governments worldwide also use auctions to allocate property rights, such as auctions for wireless spectrum [101] (with worldwide proceeds exceeding US $100 billion by the end of 2001 [105]). In a typical week in February, 2006, the U.S. treasury sells more than US $25 billion in three-month treasury bills through a sealed-bid auction.[1] Sponsored search auctions drive over $1 billion in revenue to Google each quarter [89], and the eBay marketplace reported a record US $44.3 billion volume in the 2005 calendar year, representing a 30% increase over 2004.

Why are auctions so popular? Trepte [151] emphasizes the role auctions play in promoting competition. Competition, in turn, provides incentives for bidders to act as 'honest brokers' of information, so that in the context of procurement the winner is the most technically efficient firm. Yet, auctions are only effective in promoting competition if they are trustworthy, with all bids treated fairly and equally and all bids are *seen to be* treated in this way [151]. In discussing the role of regulation in the context of procurement auctions, Trepte emphasizes the importance of being able to commit to an objective process, so that

> "... the buyer binds himself in such a way that all bidders know that he will not, indeed cannot, change his procedures after observing the bids, even though it may be in his interest to do so."

Schelling [139] had already noted "*... it is a paradox that the power to constrain an adversary may depend on the power to bind oneself.*" In the context of auctions the point is a simple one: the firm engaged in procurement would like to commit not

[1] Generally sold in uniform-price auctions. See `http://www.publicdebt.treas.gov`

to advantage one firm over another to promote fair competition.

5.1 Motivation: The Problem of Corruption

(We thank David Parkes for his extensive contributions to this section.)

Auctions are not immune to corruption and this commitment to a correct process can be hard to achieve. By *corruption*, we mean the auctioneer breaking the rules of the auction in favor of some bidder(s), typically in exchange for bribes [90]. In its procurement guidelines, The World Bank defines a *corrupt practice* as

> "... the offering, giving, receiving, or soliciting, directly or indirectly, of any thing of value to influence the action of a public official in the procurement process or in contract execution."

When an auction is used for the procurement process then the auctioneer is the person of trust. The possibility of corruption exists in an auction whenever the auctioneer is not the owner of the goods for sale in the auction, or the owner of the firm that is seeking to procure goods [90]. For instance, there is a possible conflict of interest when the auction is operated by an individual within a large firm, or by a public servant within a government organization [88].

As evidence of the extent of concern about corruption in competitive processes, the main goal of governments and international bodies such as the World Bank, in regulating public procurement auctions, is to "*curb the discretion*" of the buyer [151]. The World Bank recently estimated the volume of bribes exchanging hands for public sector procurement alone to be roughly US$200 billion per year, with the annual volume of procurement projects 'tainted' by bribes close to US$1.5 trillion [149], and has made the fight against corruption a top priority [47].

When price is the only factor in determining the winner of an auction, then many authors argue that using an open and verifiable, *sealed-bid* auction should help to prevent corruption [151, 134, 91]. In a sealed bid auction, bids are committed during the bidding process and then opened simultaneously by the auctioneer and the rules correctly followed to determine the winner (and price). However, it seems difficult in practice to ensure a fully trustworthy sealed-bid auction. The kinds of manipulations that are possible in a first-price sealed-bid auction include the following:

- The auctioneer allows a favored bidder to improve on the bid of the winning bidder (possibly the same favored bidder) by revealing information about other submitted bids before the auction closes [91], or by inserting a bid for the favored bidder after reviewing the submitted bids. This allows the favored bidder to win at the best possible price.

- A favored winning or second-place bidder can be invited to change a bid after the auction has closed in order to obtain a better price or win the auction, respectively [103].

- Bribes can can be received before bids are made, in exchange for a promise to modify the bidder's bid to the bidder's advantage should that bidder be the winner [83].

Each of these manipulations relies on the ability to circumvent the intended sealed-bid auction process. The first method relies on learning information before the close of the auction, or being able to insert or modify a bid after some bidders have already bid. The second and third methods rely on being able to change, or cancel, bids after

the close of the auction.

More than ethically troubling, corruption is undesirable because it can lead to both an efficiency loss (e.g., with the wrong supplier winning a contract) and also a distributional effect (e.g., with the government paying too much for a contract) [11, 47, 83, 103, 43, 38, 39]. Corruption is a widespread, real-world problem, as illustrated by the following examples:

- A 1988 U.S. investigation, *Operation Ill Wind*, into defense procurement fraud resulted in the conviction of 46 individuals and 6 defense corporations, with fines and penalties totaling US$190 million [38].

- The construction of subways in Italy cost US$227 million per kilometer in 1991; after anti-corruption actions the cost fell to US$97 million [38].

- Germany's auditor estimated the government suffered costs resulting from corruption in the construction field of about DM 5 billion a year [38].

- Mafia families in New York City would sometimes pay bribes for an "undertaker's look" at the bids of other bidders before making their own bids when bidding for waste-disposal contracts [83].

- A Covington, KY developer was shown the bids of two competing developers for a US$37 million courthouse construction project [83].

- In 1999, the winner of an auction for the construction of a new metropolitan airport in the Berlin area changed its bid after it acquired the application documents of the rival bidder [90].

- In 1996, Siemens was barred from bidding in public procurement auctions in Singapore for five years because they bribed the chief executive of Singapore's public utility corporation in exchange for information about rival bids [90].

- As many as 40–50 "information brokers" (buying information from oil companies and selling to suppliers) may be actively working at any given point of time in the North Sea oil industry, with corruption and bid rigging affecting upwards of 15% of contracts (an economic value of GB£1.75 billion per year in 1995) [10].

Driving home the difficulty of implementing truly sealed-bid auction processes, Ingraham [75] provides a remarkable account of corruption in New York City School Construction Authority (SCA) auctions, an approximately US$1 billion per year market. Two SCA employees and eleven individuals within seven contracting firms were implicated in the corruption. A dishonest contractor would submit a bid well below the projected price of the contract, and during the public announcements of the bids, the auctioneer would save the favored bid until the other bids were opened and announced. Knowing the current low bid, the dishonest auctioneer would then read aloud a false bid just below the current low bid instead of the artificial bid actually submitted. The bid form would subsequently be modified with correction fluid.

Second-price auctions are robust against all three of these manipulations [103]. In a second-price (Vickrey) auction the good is sold to the highest bidder for the second highest bid price [153] (respectively, bought from the lowest bidder for the second lowest bid price in a reverse auction such as a procurement auction). In a second-price auction no single bidder can be given a special advantage because all bidders

have the same opportunity to match other bids via the auction rules. However other (more complicated) manipulations are possible; e.g., the auctioneer can collude with the two highest bidders, with the second highest bidder invited to *withdraw* her bid upon the auction closing so that the highest bidder wins the auction but has to pay only the third highest bid [91].[2]

Moreover, without additional assurances, second-price auctions are vulnerable to a new kind of manipulation: when selling an item, an agent acting for the seller can insert a *shill* bid below the highest bid after the close of an auction and drive up revenue.[3]

An almost universal conclusion of the published research in the field is that there is a need for *verifiably correct* and *trustworthy* first-price sealed-bid auctions [91, 151, 29], with emphasis placed on the need for the process to be *open and transparent*. It is apparent from the above examples that standard solutions, which rely on a well-defined and open process, with bids sealed until opened in public, and the use of regulations and penalties, often remain inadequate. Indeed, Andvig [10] makes the interesting point that even when an organization is *successful* in restricting access to information before an auction closes, then, paradoxically, there are fewer people that

[2]Moldovanu and Tietzel [106] provide a remarkable account of a failed attempt by the German author Goethe (1749–1832) to use a second-price auction to sell a manuscript. Goethe set a reservation price p and instructed his agent to collect a bid b from Vieweg (1761–1835), the propsective publisher, and to sell at p if and only if $b \geq p$. The story is relevant here because his agent and legal counsel Böttiger, deviated from the rules and revealed to Vieweg the exact amount p. Vieweg subsequently bid p, and Goethe accepted the offer but without realizing his desire, which was to learn about his true "worth" by running this truthful auction.

[3]Seeing problems with implementing truly sealed-bid auctions, one can also consider the role of open auctions in which bids are "broadcast" to all participants; traditionally, this would occur with all bidders in the same room but today an open auction can be conducted over the Internet. Although open auctions may provide transparency and reduce opportunities for manipulation, Lengwiler and Wolfstatter [91] conclude the open auctions may not be desirable for the fear of bidder collusion. Other authors argue that open auctions are often unsuitable for procurement, and other complex environments, because bidders need time to formulate technical proposals [13, 90].

know enough to "police" the process and this can lead in turn to *more* opportunities for corruption.

5.2 Our Solutions

Our first solution, due to Parkes, Rabin, Shieber, and Thorpe, ensures the correctness of a sealed-bid auction for any number of identical goods and allows verifiability of correctness by any third party, without revelation of the bids received. The solution includes all popular variants of auction pricing rules, including first-price, second-price and generalized Vickrey auction (GVA) payments. Correctness is ensured by providing complete secrecy of bids until the close of the auction (including, even, from the auctioneer), assured revelation of bids to the auctioneer upon auction closing, and verification that the outcome (or the part of the outcome that the auctioneer promises to verify) is correct through the use of cryptographic methods. None of the aforementioned manipulations of first-price or second-price auctions is possible in our scheme.

Our second solution, a cryptographic combinatorial clock-proxy (CCCP) auction, due to Parkes, Thorpe, and Rabin, ensures the correctness of a clock-proxy auction advanced by Ausubel et al. in [15]. Far more general than the first solution, this computational mechanism provides for winner determination and price discovery in a combinatorial auction setting, where bidders may bid on various bundles of one or more distinct goods for sale. Cryptography provides both secrecy and provable correctness: during the auction protocol, not even the auctioneer learns any private information until the bids are closed and he can no longer influence the outcome;

after the protocol, the auctioneer must prove his actions correct using encrypted information collected from bidders during the initial phases of the auction.

An important factor in the practicality of cryptographic methods for providing trusted auctions is having a clearly understandable and convincing solution that is accessible to knowledgable people who are nevertheless not experts on the intricacies of cryptography and general zero knowledge proofs. In this regard, we assume a public key infrastructure under which all parties possess public/secret key pairs for digital signatures and private communications and illustrate our protocols with Pascal Paillier's *homomorphic encryption* [114] scheme, which provides verifiable correctness and trustworthiness without revealing information about the bids. This model and a Paillier-based implementation of it are respectively described in Chapters 2 and 3. Our cryptographic proofs are based on universally accepted assumptions.

We focus on two additional aspects of practicality. First, the auction will clear in reasonable time and with reasonable communication requirements using commodity hardware, even for a large number of bidders. Second, the computational architecture must be consistent with practical business models. To achieve this we focus on *proofs of correctness* rather than secure computation. Unlike previous solutions, e.g., Naor et al. [110], and Suzuki and Yokoo [158] (see also the literature review in Section 6.1.1), we require neither the existence of multiple auctioneers nor that the auctioneers and/or bidders collaborate to conduct the auction. We believe that a model involving a single auctioneer that is solely responsible for conducting the auction and independent verification of the auction by third parties is more realistic from a business perspective.

We have carefully examined the role of all parties in cryptographic auctions formalized their roles in a cryptographically sound protocols. In addition to a seller, multiple bidders, and an auctioneer, our models employ two commercial entities: *notaries* protect bidders by acting as witnesses to the submission of bids—primarily to prevent the auctioneer from ignoring or modifying submitted bids, and a *Time-Lapse Cryptography Service* (Chapter 4) provides a cryptographic commitment protocol that prevents bidders from refusing to reveal commitments they make during the auction protocol. The Time-Lapse Cryptography (TLC) Service is used to keep bids secret before the close of the auction. The TLC service publishes a public key before the auction begins, and delays the creation of the corresponding secret decryption key until after the close of the auction.

Whereas earlier methods required the auction to be distributed across the computers of multiple, independent auction operators, or required complex interactive protocols involving computation by bidders and the auctioneer, our solution has a simple, non-interactive, and familar computational architecture. Bidders prepare commitments to their bids and send the commitments to the auctioneer and any witnessing notaries. The auctioneer opens the commitments (but can do so only after the auction closes), determines the outcome of the auction and publishes proofs of its correctness. In return for this simplicity, we do not achieve all of the same privacy guarantees as earlier solutions [86, 110, 72, 94].

We choose not to use cryptography to completely protect against the revelation of bid information by the auctioneer *after* the close of the auction. We consider this kind of manipulation to be less dangerous because it does not facilitate corruption

during the auction. No information can be leaked by any party before the auction closes, and after the auction closes no new bids can be introduced and no bids can be altered. We also note that even when bid values stay concealed from the auctioneer at great process complexity cost, a determined adversary can try to spy on a rival and obtain information on her bid using corrupt insiders. Thus, an absolute guarantee of secrecy is never attainable in real life.

While an important area of research, the cryptographic protocols currently available for enforcing end-to-end secrecy are, in our view, too cumbersome and challenging to understand to find wide business applicability. In particular, existing solutions maintain complete secrecy of private data by sharing these data among multiple parties who cooperate throughout the computation, and are trusted not to collude. When these multiple parties are the bidders in an auction, the protocol becomes unwieldy; when they comprise a distributed auctioneer, that distributed auctioneer is functionally a trusted third party. Instead of attempting to achieve perfect security through cryptographic means, we appeal to hardware and systems solutions to prevent *ex post* information leakage and employ cryptography to preserve secrecy before the computation is fixed and prove its output correct.

Complete post auction-closing secrecy can be enforced, in cases where it is deemed essential, by appeal to specialized hardware and monitoring software. A *Trusted Computing* infrastructure, based on secure hardware and digitally signed software (audited by third parties for correctness), installed in physically secure locations with ongoing monitoring and auditing, can prevent the leaking of information with high assurance [142]. In fact, with such deliberately opaque servers it is of the utmost

import that an auction participant can independently verify the correctness of the outcome of an auction, and not rely on blind trust that the servers' programming was free of any bugs that might yield incorrect output. Thus, such technological methods to eliminate secrecy leaks are very well complemented by our methods for verifiable correctness.

While providing the secrecy of bid information is our primary focus, privacy of bidder identities can be accomplished by other business or cryptographic protocols. For example, bidders may use legal proxies to place bids on their behalf to hide their identity, or the auctioneer may employ a cut-and-choose mixing technique (as described in Section 7.2.1) so that the mapping of winners to bidders is revealed only where necessary by revealing the random re-encryption factors.[4]

To demonstrate the scalability of this technology, we conducted empirical timing tests (Section 6.5) using commodity computing hardware common in 2006. We show that for acceptable strength of the cryptographic security key, single or multi-item auctions with 100 bidders can be prepared in around two hours of computation and verified in less than half an hour, all on a modest (2.8 GHz Pentium 4) PC. We also show that the computations scale linearly with the number of bidders. Because our method is easily parallelizable, it is possible to accommodate auctions with even tens of thousands of bidders in at most a day of computation on a 64-node network of commodity PC's. Over a decade ago, Franklin and Reiter [65] also found that conducting cryptographic sealed-bid auctions was possible on commodity computing hardware of the day, although their protocol differs substantially from our own.

[4]This point becomes important when proving the outcome of the auction; in the protocols we describe, we do not attempt to keep secret that, say, bidder B_3 was the winner, because we assume B_3's true identity is already private if that is necessary.

5.3 Additional Benefits: Better Robustness to Collusion

Providing for verifiable and trustworthy auctions *without* revealing information about bids brings another indirect benefit. A major concern in the use of auctions in practice is that of *bidder collusion* [136]. By collusion we mean bidders coordinating in a *bidding ring*, with the intention to manipulate the final price. The basic idea is to bid jointly in order to limit competition, with the proceeds being shared among members of the ring.[5]

Collusion between bidders is an especially difficult problem to address because it necessarily exploits information asymmetries between the auctioneer and the bidders, and is therefore hard to prevent and detect [151]. Unlike the recommendations of the World Bank and other national and international agencies, our technology allows for auction verification without revealing information about bids, and this provides further robustness against bidding rings.

As evidence of the problems caused by bidder collusion, consider the following examples in first-price sealed-bid auctions:

- Multiple firms were convicted of participating in bidding rings in auctions for school milk contracts in Florida and Texas in the 1970s and 1980s [123].

- Following allegations of bidder collusion at Forest Service timber sales in the Pacific Northwest in the 1970s, an empirical study finds evidence for collusion

[5]Porter and Zona [125] note that joint bidding is typically illegal unless the specified work could not be performed without the combined capabilities of the participating firms or if the bidders could not be competitive individually.

in auctions conducted between 1975 and 1981 [19].

- In 1984, one of the five biggest highway construction firms in New York state was convicted in federal court of rigging bids in auctions for state highway contracts on Long Island in the early 1980s. Four other firms were listed as unindicted co-conspirators [125].

First-price auctions are preferred over second-price auctions because they are less susceptible to collusion.[6] In first-price auctions, bidding rings are only sustained by the threat of punishment because members have to submit bids lower than their true value. Bidding rings are unstable without the ability to identify a bidder that deviates and without repeated interaction [133, 71, 75]. Indeed, Ashenfelter [12] suggests that auction houses such as Sotheby's and Christie's keep the identity of buyers secret to combat rings so that buyers can break from a ring and buy anonymously.

Yet, a common feature in every one of the aforementioned real-world auctions was that the auction was concluded with the *public opening* of bids. As discussed by Porter and Zona [125], this has an unfortunate side effect:

> "The... policy of publicly announcing the bids and the identity of bidders allows cartel members to detect deviations from cartel agreements. Undercutting or cheating would not go unnoticed."

Indeed, the World Bank's own official *Procurement Guidelines* [149] state that the World Bank

> "Bids shall be opened in public; bidders or their representatives shall be allowed to be present... The name of the bidder and total amount of each

[6] In a second-price auction the collusive strategy is for the member of the ring with the highest value to bid high and the rest of the members to bid low, or not at all. This is stable because no member of the ring can do better through a unilateral deviation from the collusive agreement [133].

> bid, and of any alternative bids if they have been requested or permitted, shall be read aloud (and posted online when electronic bidding is used)..."

Why, one might ask, is bid information made public when it can enable bidding rings to sustain themselves through credible threats of punishment? Trepte [151] makes the reason very clear. While noting the value of "*restricting the detail and content of post-award information*," he adds that "*the existence of such information is essential if disappointed buyers are to be able to challenge unfair or unlawful procurement procedures.*" For this reason, we argue that our solution may have important ramifications in terms of reducing opportunities for bidder collusion while addressing corruption. Our auction protocol provides a balance of transparency, trustworthiness and secrecy that reduces the potential for corruption while improving market efficiency.

A related point can be made in the context of using our techniques to verify the correctness of second-price auctions. The main effect, of course, is that we enable a trustworthy and verifiabily-correct auction process. This prevents, in particular, any concern about the manipulation through shill bidding discussed earlier. But there is also a second benefit, that comes from not needing to reveal bid values in establishing that the auction process was correctly conducted. Second-price auctions support truthful bidding in a dominant strategy equilibrium, usefully simplifying the bidding process for participants. On the other hand, this bidding strategy can have a number of unpleasant side effects when bids are revealed after the auction closes. In the context of procurement, a supplier will be reluctant to reveal her true cost basis to a competitor [135]. Similarly, when purchasing government assets such as wireless spectrum, a bidder will be recluctant to reveal her true value for acquiring assets

to competitors. Governments may also be reluctant to reveal to the public that the value of the highest bid was significantly more than the revenue collected.[7]

[7]For example, when the New Zealand government conducted a Vickrey auction for telecommunications licenses, it was revealed after the fact that the winner had been willing to pay much more [101].

Chapter 6

Practical, Secrecy-Preserving, Provably Correct Sealed Bid Auctions

6.1 Introduction

In this chapter, we present a practical protocol for sealed-bid auctions that prevents the manipulations described in Chapter 5. An important factor in its practicality is having a clearly understandable and convincing solution accessible to knowledgable people who are nevertheless not experts on the intricacies of cryptography and general zero knowledge proofs. To that end, we have carefully examined the role of all parties in a sealed-bid auction and formalized their role in a cryptographically sound protocol.

We deal with the real-world issues that arise in the actual implementation of such a

system. We define comprehensive security goals for our auctions. All bid information must be secret from everyone, even the auctioneer, until the auction closes. After the auction closes and results are computed and announced, the only information that is revealed either states the outcome as defined in the auction rules, or information implied by that outcome.[1] At the same time, a proof of correctness of the results is published by the auctioneer, allowing anyone to individually verify that correctness.

We also enforce that the auctioneer include all properly submitted bids. Finally, even though bids are submitted by participants in concealed form, our system enforces revelation of bids to the auctioneer after the closing time of the auction. To achieve our goals we rely only on well known, universally accepted cryptographic assumptions for security before the auction's close, and explicitly do not trust the auctioneer or any other party during this phase of the auction. In addition to a seller, multiple bidders, and an auctioneer, our model employs two commercial entities: *notaries* protect bidders by acting as witnesses to the submission of bids, and a *Time-Lapse Cryptography Service* [129] provides a cryptographic commitment protocol that prevents bidders from refusing to reveal commitments they make during the auction protocol.

We assume only commodity computing resources and a public key infrastructure under which the auctioneer, seller, bidders, and notaries all possess public/secret key pairs for digital signatures. Pascal Paillier's *homomorphic encryption* [114] scheme is used to provide verifiable correctness and trustworthiness without revealing information about the bids. The Time-Lapse Cryptography (TLC) Service is used to keep bids secret before the close of the auction. The TLC service publishes a public

[1]For instance, in a second-price single-item auction, the second highest bid price is revealed because this is implied by the outcome.

key before the auction begins, and delays the creation of the corresponding secret decryption key until after the close of the auction.

While secrecy of bid information is our primary focus, privacy of bidder identities is not a goal of our work and can be accomplished by other business or cryptographic protocols. For example, bidders may use legal proxies to place bids on their behalf to hide their identity, or the auctioneer may employ a cut-and-choose re-encryption technique as described in Section 6.2.4 so that the mapping of winners to bidders is revealed only where necessary by revealing the random re-encryption factors. This point becomes important when proving the outcome of the auction; in the protocols we describe, we do not attempt to keep secret that, say, bidder B_3 was the winner, because we assume B_3's true identity is already private if that is necessary.

We focus on two aspects of practicality. First, the auction must clear in reasonable time and with reasonable communication requirements, even for a large number of bidders. Second, the computational architecture must be consistent with practical business models. To achieve this we focus on *proofs of correctness* rather than secure computation. Unlike previous solutions, e.g., Naor et al. [110], we require neither the existence of multiple auctioneers nor that the auctioneers or bidders collaborate to conduct the auction. We believe that a model involving a single auctioneer that is solely responsible for conducting the auction and independent verification of the auction by third parties is more realistic from a business perspective.

We have chosen *not* to explicitly protect against an auctioneer revealing bid values and quantities after an auction has closed and the outcome has been announced; in Section 5.2, we appeal to Trusted Computing and similar systems technologies to

protect against such attacks.

To demonstrate the scalability of our technology, we have conducted preliminary timing tests (Section 6.5). We show that for acceptable strength of the cryptographic security key, single or multi-item auctions with 100 bidders can be prepared in around two hours of computation and verified in less than half an hour, all on a standard (2.8 GHz Pentium 4) PC. We also show that the computations scale linearly with the number of bidders. Because our method is easily parallelizable, it is possible to accommodate auctions with even tens of thousands of bidders in at most a day of computation on a 64-node network of commodity PC's.

6.1.1 Related Work

Much of the previous work on the use of cryptography for conducting verifiably correct and trustworthy auctions has focused on the goal of *complete privacy*, where not even the auctioneer learns information about bids after the close of the auction [86, 110, 72]; see Brandt [32] for a recent discussion. This is typically achieved through assuming two or more trusted third parties, either through numerous auctioneers [72] or with asymmetric models in which the commerical entity of an *auction issuer* is assumed in addition to the auctioneer [110, 94]. Some protocols achieve complete privacy through bidder-resolved multi-party computation [32]. In comparison, we settle for verifiable correctness and trustworthiness in combination with complete secrecy to all parties except the auctioneer; see also Franklin and Reiter [65], which employs "verifiable signature sharing", requires an electronic cash infrastructure, and distributes this trust in the auctioneer among a set of servers. As discussed above, the

auctioneer in our solution cannot learn any information about bids until the auction has closed. In return we achieve a non-interactive[2] protocol that is especially simple from a bidder's perspective.

In justifying the focus on computationally secure methods to provide correct and verifiable auctions, it is interesting to note that achieving information-theoretic guarantees on complete privacy is impossible in a single-item Vickrey auction [34], at least when it is desired that the payment is only revealed to the winning agent. (One cannot prove to another party that the winner's payment was correct without revealing information beyond that implied by the fact that this bidder had the highest bid.)

For trusted third parties we require only notaries, who provide a lightweight "witness" service and are independent business entities that already exist in practice [145]. The level of trust in them is quite low, as they never possess any nonpublic information. The Time-Lapse Cryptography Service functions as a trusted third party, although Rabin and Thorpe [129] describe a TLC service that distributes trust among many parties using secret sharing, so that there is no single completely trusted party and reconstruction of the decryption key is robust to node failures.

In addition to providing business realism (also see Lipmaa et al. [94] for a critique of earlier methods), we choose to adopt standard methods from homomorphic encryption and eschew more complex cryptographic ideas such as secure multi-party computation, obfuscation of circuits, and oblivious transfer. As Bradford et al. [31] argue, many such complex protocols, particularly those requiring the ongoing par-

[2]Interactive cryptographic auction protocols require the active participation of bidders throughout the auction process in order to obtain the auction results, generally via multi-party computation or related methods. Non-interactive protocols such as ours require no such bidder participation; submission of bids is the only required bidder activity, and bidders' verifications of auction correctness can be performed with no additional interaction with the auctioneer.

ticipation of bidders, suffer from "protocol completion incentive problems", in which bidders who know they have lost or change their minds can disrupt the protocol and prevent the completion of an auction. We intentionally avoid such problems by having a single partially trusted auctioneer compute the outcome.

We share with Lipmaa et al. [94] (see also [6, 21, 32, 147, 45]) the use of homomorphic encryption, but seek a simpler solution through the use of a single auctioneer in place of the two server model adopted in their work. In their protocol, the seller and an auction authority, who are trusted not to collude, work interactively to generate zero-knowledge proofs of correctness. Cachin [41] proposes a technique based on homomorphic encryption in which a semi-trusted single auctioneer provides a means for two bidders to determine whose bid is higher in zero knowledge (in fact, not even the auctioneer learns the bids). However, his extended protocol for cryptographic auction similarly requires two auction servers which are assumed not to collude. Nakanishi et al. [108] describe a similar protocol based on additively homomorphic encryption and a set of auction servers who conduct a multi-party computation. Such methods result in stronger privacy and secrecy properties at the cost of this additional process complexity.

Rabin, Servedio and Thorpe [128] have recently proposed a somewhat different cryptographic architecture suitable for conducting sealed-bid auctions with similar properties that does not employ homomorphic cryptography. Instead, the system uses a statistically secure encryption scheme based on cryptographic commitments and proves all computations correct to an arbitrarily low probability of error.

Earlier work on multi-item auctions either assumes distributed trust [82, 45, 6],

or adopts multi-party computation techniques [32], and the current state of the art for secure combinatorial auctions is still not very scalable [157, 147]. In comparison, our approach can be extended to secrecy-preserving multi-item auctions (presented here) and combinatorial auctions (Chapter 7). Specifically, our trusted auctioneer can apply fast algorithms to the combinatorial optimization problem in determining winners. The auctioneer must simply construct a *proof* that the outcome is correct and need not involve multiple parties in *computing* the outcome.

Whereas previous architectures use cryptography for anonymity, we note that existing real-world business entities (e.g., notaries as proxy bidders) also meet this need. We therefore do not complicate our protocol with maintaining bidder anonymity and consider it outside the scope of this work. Another practical issue, addressed in previous work but not here, is that of *noncoercibility* [40, 145] of an auction. Noncoercibility prevents a bidder from being able to credibly claim to a third party that it bid in a particular way after the close of an auction. Auctions with this property are more resistant to bidding rings, since the stability of bidding rings in first-price auctions depends on being able to detect (and punish) deviations from agreed upon rules.

6.2 Preliminaries

The standard auction model considers an auctioneer AU, bidders $B = \{B_1, \ldots, B_k\}$, and a seller. This is a *forward* auction in that the goal is to allocate one or more items to some set of bidders. Reverse auctions, with a buyer rather than a seller, are suitable for procurement auctions and can be modeled in a similar

way. In a single item auction, each bidder B_i is modeled with a private value v_i; she bids to maximize her net utility (which is $v_i - p$, her payment, in the event that she wins the auction.) In a first-price, sealed-bid auction, each bidder B_i makes a bid $\mathbf{Bid}_i$. This is a claim about her maximium willingness to pay. Bids are made without any information about the bids (or values) of other bidders, and the item is sold to the highest bidder, who pays the highest bid price. In a second-price sealed-bid auction, the item is sold to the highest bidder, who pays the second highest bid price.[3] See Krishna [85] for an introduction to auction theory.

6.2.1 Desired Auction Properties

Based on the analysis in the introduction, we list desiderata for any sealed-bid auction process. These go beyond the standard economic goals, for instance, efficiency or revenue maximization:

- Non-repudiation by bidders: Once a bidder submits a bid, her bid is provably unalterable. Moreover, a bidder is bound to reveal her bid to the auctioneer after the auction closing time.

- Non-repudiation by auctioneer: The auctioneer's exclusion of a properly submitted bid can be conclusively proven and thus becomes legally actionable.

- Trustworthiness: The auctioneer cannot know the bids until after the close of the bid submission phase. Thus the auctioneer cannot collude with bidders by sharing others' bids during the auction.

[3] As noted earlier, although more susceptible to collusive bidding behavior, second-price auctions have the useful property that it is a dominant strategy for a bidder to report her true value.

- Secrecy: The bids are hidden to everyone until all bids are committed. At the close of the auction, only the auctioneer knows any secret information. He may keep the outcome secret, notifying only winners of their allocations and payments, or make any part of the outcome public by revealing some or all of the allocations and payments and proving them correct. Revelation of these values does not reveal other secret information not implied by the values themselves.

- Verifiable correctness: All information revealed, whether private or public, is proven correct. Bidders receive a proof of the correctness of their own allocation and payments. The public, including all bidders, receives a proof of correctness for all public information about the outcome of the auction and also the validity of bids. The auction protocol enforces correctness; an auctioneer will not be able to present valid proofs for invalid winners or incorrect payments.

In achieving these properties we make standard cryptographic assumptions. Because the security of our encryption is related to the computational intractability of solving "hard" cryptographic problems, longer cryptographic keys can be adopted over time as computational hardware gets more powerful. This will maintain the same level of realized security at comparable computational running time.

6.2.2 Real-World Components

We recall that our auction system comprises an auctioneer AU, bidders $B = \{B_1, \ldots, B_k\}$, and a seller. Bidders can also be *proxies* to provide anonymity. In addition, we assume a universally accessible, tamper resistant clock (such as provided by the United States NIST time servers) and the following components.

Certified Bulletin Board

The auctioneer maintains a certified bulletin board. This can be a publicly known website maintained and updated by the auctioneer. The auctioneer uses the bulletin board to post all public information about the auction, including the initial auction announcement as well as (encrypted) information about bids that have been submitted and proofs that can be used to verify all publicly available information about the outcome. All posts to the the bulletin board will carry appropriate digital signatures identifying their originators.

Notaries

Notaries are reputable agents, such as law firms, accountants, or firms specializing in providing a *witness* for bidders. When preparing to participate in an auction, a bidder may select a set of notaries of her choosing from some set of notaries possibly authorized by the auctioneer. Use of the notaries is optional; their only purpose is to prevent a dishonest auctioneer from failing to post bid information from disfavored bidders. In using a notary, whenever a bidder sends concealed bid information to the auctioneer she also sends that concealed information to any notaries she has selected, most notably commitments to bids and random help values. These notaries also submit this information to the auctioneer, and act as witnesses in the case that a bidder complains that an auctioneer does not correctly post her information to the bulletin board. We require that a majority of the notaries is not corruptible. Note that our process is structured so that no information about the actual bids is revealed to the notaries, and, again, their only role is to serve as witnesses to the communications

in the auction in case of a dispute between a bidder and the auctioneer.

Time-Lapse Cryptographic Service

A bidder B_i, possibly in collusion with the auctioneer, might refuse to open her commitment and reveal her encrypted bid $E(\mathbf{Bid}_i)$.[4] One way to prevent this practice of *bid repudiation* is to employ the "Time-Lapse Cryptography Service" named and described by Rabin and Thorpe in Chapter 4 and [129].

The Service will at regular intervals post a new cryptographic public encryption key *TPK* (Time-lapse Public Key), and after a fixed period of time post the associated secret decryption key *TSK* (Time-lapse Secret Key). For our purposes, it suffices that the public key be available before the bids are to be submitted, and that the secret key be released soon after the auction closes. We envision for the purposes of this chapter that the Service will publish a constant stream of keys with appropriate lifespans for an auction, and the Auctioneer selects and specifies a key to be used that expires soon after the closing time of the auction. For example, the Service might publish a set of public encryption keys each hour, each with a different lifespan, e.g. three hours, one day, one week, 90 days, etc. When the lifespan expires for a particular public encryption key, the Service reconstructs and publishes its associated secret decryption key.

For our purposes, the Service must not employ any single trusted third party who knows either the bid information or any secret key that could decrypt the encrypted bid information before the close of an auction. Additional relevant cryptographic

[4]The notation $E(m)$ designates an *encryption* of a message m; see Chapter 2 for details of the cryptographic notation we employ.

details about the TLC Service are provided in Section 6.2.4, and a full treatment is found in Chapter 4.

6.2.3 Overall Flow and Main Steps of Auction

Schematically, the auction process will proceed in three main stages (described in more detail in Section 6.3). In the first stage, the auctioneer posts the auction announcement on the bulletin board. The announcement, to be detailed later on, includes a deadline time T for submitting bids. In the second stage, the bidders commit to the encrypted forms of their bids and random data but post bid information in a form that is concealed even from the auctioneer. Notaries are engaged in this stage and witness these commitments posted to the auctioneer's bulletin board. In the final stage, the bidders must follow through and reveal the encrypted forms of their bids to the auctioneer and the public. They do *not* decrypt or reveal their unencrypted bids. The auctioneer and other bidders verify that these encryptions of their bids are consistent with the posted commitments. The auctioneer then decrypts the bids in secret, and computes the outcome of the auction according to the posted rules for that auction. He then posts the parts of the outcome to be verified on the bulletin board, along with public proofs that the selection of the winner(s) and their payments was done according to the auction rules. After the last posting, any party can verify the correctness of the publicly verifiable part of the outcome. A bidder can also privately verify the correctness of her individual outcome via a proof offered by the auctioneer if that outcome is to be kept secret.

6.2.4 Basic Cryptographic Tools

Our system relies on universally accepted cryptographic tools. We describe the tools we employ in our result, referring to other publications for established results and providing proofs for new uses of existing tools. We will sometimes refer to a "prover" $\mathcal{P}$ and a "verifier" $\mathcal{V}$ when discussing the secrecy-preserving proofs of mathematical facts relating to our auctions. See the *Handbook of Applied Cryptography* [102] for a general introduction to the applied cryptographic techniques and notation we employ.

Public Key Infrastructure

We assume cryptographically sound methods of establishing and exchanging public keys used for all the cryptographic tools we employ, including the auctioneer's public/secret key pair for Paillier encryption and the public and secret keys published by the time-lapse cryptography service. In addition, the auctioneer, notaries, and all bidders require public/secret key pairs for digital signatures. The public signature verification keys of all parties must be mutually known and certified. We notate digital signatures as follows: AU can sign message x, generating $\mathbf{Sign}_{AU}(x)$. A bidder B_i's signature of x is denoted $\mathbf{Sign}_i(x)$.

Sources of Randomness

Cryptographic key generation and probabilistic encryption require a good source of random data. We postulate bidders' and notaries' ability to create enough highly random data to create strong key pairs, encrypt and sign a small number of values, and generate the secure random data string we introduced in Section 2.1.3 and recall below. Such a source might be hardware that extracts randomness from radio static

or quantum noise in diodes. Such "hardware randomness generators" are already employed in important cryptographic applications.

Secure Random Data

In order to prevent any party (including the auctioneer) from cheating or engaging in steganographic communications with the outside by infusing deliberately tainted random data into the cryptographic protocols, we require that all bidders commit to a random data string when they bid, and that the auctioneer post a commitment to a random data string when posting the auction rules. These strings are revealed only at the close of the auction, and then combined using exclusive OR so that even if just one of the strings is truly random, the combination thereof is also truly random. We denote this auction random data string by σ. The resulting string σ is used to "tie the hands" of the auctioneer: when proving the correctness of the auction, the auctioneer must reveal data exactly as specified by the bits in σ.

The auctioneer publishes the algorithms to be used on data from the random data string in the auction rules, for example, the method for choosing a random permutation of integers in a specific range, which is employed in the course of proving the auction results correct.

Time-Lapse Cryptography

The Time-Lapse Cryptography Service (introduced above as a real-world entity and formalized in Chapter 4) provides for a binding and hiding commitment to bids (so that the bidder may not change her bid and the auctioneer learns nothing about the bid from its commitment); it also enforces the nonrepudiation of bids, so that

once a bidder has committed to her bid, she may not prevent the auctioneer from eventually decrypting it. We assume the TLC service wherever we employ cryptographic commitments in our protocol, and notate bidder B_i's commitment to a value x as the time-lapse encryption $E_{TPK}(x)$.

Each bidder B_i commits to her encrypted bid by encrypting $Z = E_{TPK}(E(\mathbf{Bid}_i))$ (where the bid is first encrypted with the public key of the auctioneer), using a time-lapse public encryption key TPK. The bidder then posts $\mathbf{Sign}_i(Z)$ on the bulletin board. After time $T+1$, the decryption key TSK associated with TPK will be posted by the TLC service. The release of the decryption key TSK will enable the auctioneer (and everybody else) to decrypt $Z = E_{TPK}(E(\mathbf{Bid}_i))$ after time $T+1$ and thus obtain $E(\mathbf{Bid}_i)$; this functions as the "decommit" operation—importantly, out of the hands of the bidder.[5]

Where time-lapse encryption of long strings is required, a symmetric block cipher key is created and encrypted using the public TLC key, then published. Data are encrypted using the symmetric key; when the TLC secret decryption key is revealed, the symmetric key can be recovered and the data decrypted. Thus the magnitude of a time-lapse encrypted value x may be polynomial in the size of the TLC key given the assumptions underlying time-lapse cryptography and any suitably secure block cipher. We therefore assume any value in our protocol may be encrypted using a TLC public encryption key.

[5]In practice, the method for time-lapse encryption of the encrypted bid should undergo a thorough cryptanalysis to identify any potential attacks. Because of the homomorphic properties (and thus malleability) of the underlying cryptosystems, we do not recommend direct ElGamal encryption of a Paillier-encrypted value.

Re-Encryption

If privacy is to be enforced by the auctioneer, or in cases where it is necessary to keep secret the number of parties allocated any items (Section 6.4.2), the Paillier encryption scheme we use permits the re-encryption of a known encryption of a value (ciphertext) into another ciphertext so that both decrypt to the same input value (plaintext). To re-encrypt a Paillier-encrypted value, say, $E(\mathbf{Bid}_i, r)$, the prover computes a random factor $s \in \mathbb{Z}_n^*$ and computes $s^n \cdot E(\mathbf{Bid}_i, r) \equiv E(\mathbf{Bid}_i, r \cdot s) \pmod{n^2}$. This remains a valid encryption of $\mathbf{Bid}_i$, but only someone who knows s or the secret decryption key ϕ can prove that fact.

Re-encryption is well-complemented by a "cut-and-choose" protocol so that a prover $\mathcal{P}$ constructs $2v$ random re-encryptions of a set of values, then the verifier $\mathcal{V}$ asks for v of the sets to be revealed by revealing the random re-encryption factors used to construct them. $\mathcal{V}$ then checks that each re-encrypted set contains exactly the original set of elements. For example, once the posted bids are on the bulletin board, the auctioneer creates a number of re-encrypted auctions, verifies half of them to be correct, and then proves the outcome of the auction on the other half. This keeps the original bidders' identities private.

The computational cost of re-encrypting a ciphertext is almost equivalent to encrypting a plaintext, because the dominant computation is the modular exponentiation required by both operations.

6.3 Single-Item Auctions

Given the general cryptographic tools developed in Chapters 2 and 3, we can now describe a single-item cryptographic auction. We assume that the bidders $B_1, \ldots, B_k$ are known entities with publicly known digital signatures $\mathbf{Sign}_i$. We further assume that the winner and her payment depend only on the ordering of the values of the bids and that the payment is one of the bids.

This class of auctions includes first-price and second-price auctions, and also allows for auctions with reservation prices by a simple extension in which the seller also submits a bid.[6] Thus, this class also includes *revenue-maximizing* auctions, as described in Myerson [107], in symmetric environments in which all bidders are assumed to have independent private values drawn at uniform from the same distribution.

For clarity, we focus here on an auction in which the complete outcome of the auction—the winner and the payment by the winner—is made public and then proved to be correct. The same techniques can be used to selectively prove part of the outcome to some party, for instance to prove the winner but not the winner's payment is correct.

6.3.1 Protocol

Step 1. The auctioneer *AU* posts the following information on the bulletin board: the terms of the auction specifying the item, the mechanism for selection of the winner,

[6] In a Vickrey auction with a reservation price, in addition to bids $\mathbf{Bid}_1, \ldots, \mathbf{Bid}_k$ there is a price *rp* from the seller which is handled just as any other bid. The item is sold to the highest bidder if the maximal bid is at least *rp* but goes unsold otherwise. (Think of this as "selling back to the seller".) When sold, the payment is the maximal value of the second highest bid and the reservation price. Note that because the seller must commit to his reservation price just like any other bidder there is no danger of shill bidding.

the deadline T, an identifier ID of the auction, and a Paillier encryption key n. AU knows the corresponding decryption key ϕ. The auctioneer also posts information about any notaries that are to be used for the auction. He posts the time-lapse encryption key TPK to be used by all participants in constructing their commitments. Finally, the auctioneer posts a commitment to his random string $\mathbf{Com}_{AU}(\sigma_{AU})$ and a specification of the method that will be used to extract random permutations from the auction's random data string σ.[7]

We emphasize that all of the above data D_{AU} is posted on the bulletin board, accompanied by AU's signature $\mathbf{Sign}_{AU}(D_{AU})$.

Step 2. Every B_i chooses a bid $\mathbf{Bid}_i$. She encrypts it as $C_i = E(\mathbf{Bid}_i, r_i)$ using the public key n and a randomly chosen help value r_i. In order to create efficient test sets to prove bid sizes, we restrict the size of the bid so that $\mathbf{Bid}_i < 2^t < n/2$ for small t, say, $t = 34$. Every B_i also generates a random bit string σ_i of appropriate length which will be used in the proof of correctness. Bidder B_i then commits to C_i and σ_i by encrypting with E_{TPK} to form a single commitment string $\mathbf{Com}_i = E_{TPK}([C_i, \sigma_i, ID])$, which also includes the auction identifier ID. Finally, the bidder signs this commitment, and sends $\mathbf{Sign}_i(\mathbf{Com}_i)$ to AU and her notaries, if used, before time T. AU returns a signed receipt $R_i = \mathbf{Sign}_{AU}([\mathbf{Com}_i, ID, T])$.

Note that hiding of the encrypted bids and of the random strings by use of the secondary encryption prevents anyone from gaining any knowledge of the data prior to time T. In particular, neither the notaries nor the auctioneer have any meaningful information.

[7] We recall from Section 6.2.4 the random strings σ_i XORed together to yield the auction random data σ. AU must specify here the method used to extract a permutation of test sets from σ before AU sees σ so that everyone knows AU is revealing a truly random selection of test sets.

Step 3. At time T, the AU posts all the received commitments $\mathbf{Com}_1, \ldots, \mathbf{Com}_k$ on the bulletin board, as well as a random bit string σ_{AU}. AU also creates a number of test sets $TS_1, TS_2, \ldots, TS_K$, where K is a multiple of k, e.g., $K = 80k$. He signs and posts the test sets on the bulletin board.

Step 4. Between time T and $T+1$, any bidder B_i who has a receipt R_i for a bid which is not posted can appeal her non-inclusion, resorting to her notaries if she has used them.

Step 5. After time $T+1$, everyone, including the auctioneer AU and all bidders B_i, can recover all encrypted bids $C_i = E(\mathbf{Bid}_i, r_i)$ as well as all random strings σ_i. This is done by employing the decryption key TSK posted by the TLC service to decrypt all the commitments posted in Step 2. After time $T+1$, AU posts the encrypted bids, $C_1, \ldots, C_k$, and the random strings, $\sigma_1, \ldots, \sigma_k, \sigma_{AU}$, on the bulletin board. Every bidder B_i can verify, for any bidder B_j, that the posted value $\mathbf{Com}_j$ corresponds to the ciphertext C_j and the random data string σ_j. In case of discrepancies she protests. This check can be performed simply by decoding the commitments as above and verifying the digital signatures on these commitments. Every interested party constructs the auction's random data string σ by combining the published strings: $\sigma = \sigma_1 \oplus \ldots \oplus \sigma_k \oplus \sigma_{AU}$.

Step 6. Using the decryption key ϕ, AU recovers the bids $\mathbf{Bid}_1, \ldots, \mathbf{Bid}_k$ for computing the auction results and associated random help values $r_1, \ldots, r_k$ for constructing the proofs of correctness. The auctioneer then computes the winner of the auction and the payment according to the auction rules. The auctioneer posts the winner's identity B_i and information defining the payment to be made by the winner

on the bulletin board. This information about payment can be posted in an encrypted form if the payment is to be kept secret from nonwinning bidders. Finally, and most importantly, the auctioneer also posts information that will enable any party to verify that the correct result was implemented. These include proofs of the correctness of the winner and payment, and proofs of the validity of each bid.

6.3.2 Verification

We now show how any verifier $\mathcal{V}$ (including any of the bidders) can verify on her own that the winner and payment of the auction were determined according to the rules of the auction. This will be done in a "zero knowledge" fashion, that is, without revealing anything about the value of any bid except that implied by the outcome of the auction. In addition, the auctioneer can choose how much of the outcome is revealed. For example, the proof can validate that an encrypted payment was correctly determined but without revealing any information about the value of the payment.

The class of single-item auctions under consideration (including first-price and second-price auctions) has the property that the winner and payment depend only on the *ordering of the bids*. Take as an example the Vickrey auction and assume, without loss of generality, that the prices posted by bidders $B_1, \ldots, B_k$ are monotonically decreasing (though there may be tied bids). *AU* announces that B_1 is the winning bidder, which is tantamount to the following set of claims:

$$\{\mathbf{Bid}_1 > \mathbf{Bid}_2;\ \mathbf{Bid}_2 \geq \mathbf{Bid}_3;\ \ldots; \mathbf{Bid}_2 \geq \mathbf{Bid}_k\} \tag{6.1}$$

Note that the encrypted values

$$\{C_1, \ldots, C_k\} = \{E(\mathbf{Bid}_1, r_1), \ldots, E(\mathbf{Bid}_k, r_k)\}, \tag{6.2}$$

were posted in Step 5 of the protocol. To prove the claims, it suffices to show that each C_i is an encryption of a valid bid $0 \leq \mathbf{Bid}_i < 2^t < n/2$ for all i, and that

$$\{C_1 \rhd C_2, C_2 \trianglerighteq C_3, \ldots, C_2 \trianglerighteq C_k\} \tag{6.3}$$

Verifier $\mathcal{V}$ verifies these $2k-1$ claims in a zero knowledge fashion using the tools described above, which enables verification of the winner, item allocation, and payment as described in the following paragraphs.

Recall that the auctioneer had posted $2k$ groups of 40 test sets in Step 3. He creates proofs for each of the first k claims using k of these groups of 40 test sets, one for each claim. He reveals all encryptions for the subgroup of 20 test sets determined by the random string σ and the random method posted in Step 1 of the auction. With each of the 20 other test sets AU performs the computation described in Section 3.2.3 (**Range Protocol**) and posts it on the bulletin board. $\mathcal{V}$ can verify that all the revealed test sets are valid, that their indices were chosen correctly, and that the k posted computations are of the form (3.8). This verifies the first k claims. In addition, AU posts proofs for the $k-1$ claims that $\mathbf{Bid}_1 > \mathbf{Bid}_2$ and $\mathbf{Bid}_2 \geq \mathbf{Bid}_i, 2 < i \leq k$ by using $k-1$ groups of 40 additional test sets for each inequality using the methods described in Section 3.2.1.

This ordering of bids is used to verify the winner as the bidder with identity corresponding to submitted bid $E(\mathbf{Bid}_1)$, and the item is allocated to this bidder. In a Vickrey auction, the payment to be made by the winner is $\mathbf{Bid}_2$ and this can

be proved by sending a verifier $\mathcal{V}$ the random help value r_2 from B_2's encrypted bid $C_2 = E(\mathbf{Bid}_2, r_2)$. $\mathcal{V}$ can then verify the correctness of its payment by re-encrypting $\mathbf{Bid}_2$ with r_2 and checking the result is C_2.

In the case of a tie, where $\mathbf{Bid}_1 = \mathbf{Bid}_2$, this can also be proven using a zero-knowledge equality proof. (Indeed, the auctioneer would not be able to prove $E(\mathbf{Bid}_1) \rhd E(\mathbf{Bid}_2)$.) Tiebreaking in the single item case is done according to the auction rules, either by conducting another auction or randomly selecting a winner using the auction random data string σ according to rules defined at the beginning of the auction.

6.3.3 Verifying Partial Information about Outcomes

As mentioned in the introduction, there exist many examples in which the public disclosure of the bids or outcome of an auction is undesirable. Because there are a number of factors that play a role in determining which data are to be revealed at the close of the auction, our system provides the flexibility for the auctioneer to prove specific facts about the bids or outcome of the auction to only the individuals who need to know, without revealing anything more.

Many real-world auctions reveal such partial information, perhaps most notably the U.S. Treasury auctions for U.S. public debt, where only partial information about the bids is revealed. In that case the reputation of the Treasury provides the trust necessary for them not to disclose complete auction information, but where such a reputable auctioneer or seller is not involved, our correctness proofs provide the trust necessary to conduct such an opaque auction.

The flexibility of our system comes from the architecture of our correctness proofs; a verifier computes mathematical operations on public values posted to the bulletin board (the bidders' encrypted bids, random strings, and auction rules); the auctioneer then reveals a small amount of special data to the verifier that they compare to their calculations to verify the proof. This allows the auctioneer to control exactly who gets a correctness proof of any fact by private revelation of that special data. We illustrate below the power of our approach by examples of various partial information the auctioneer might reveal about bids and payments.

Bids. At one extreme, the auctioneer can reveal all bids to the public by revealing the random help values used to encrypt the bids. At the other, the auctioneer need not reveal any bid to any bidder to prove the payments correct. Yet there may be legal or auction theoretic reasons to provide "partial transparency" of bids. Due to the nature of the homomorphic cryptosystem employed, the auctioneer can reveal interesting partial information about the bids that can be computed using linear functions of the bid values. For example, the auctioneer might wish to reveal only the mean bid—equivalent to the sum of all bids, assuming the number of bids is public. He does this by revealing the random help value required to decrypt the product of all encrypted bids (which is an encryption of the sum of all bids). The auctioneer could also reveal other interesting statistics, such as the maximum and minimum bid, the median bid, the mean of the bids excluding the highest and lowest bid, or even the standard deviation of bids.

Payments. The auctioneer may also prove winners' payments correct in a public or private fashion. For example, instead of revealing winners' payments to everyone,

each bidder can act as her own verifier. She computes an encryption of her payment on her own, and then decrypts it with the auctioneer's help. The auctioneer can privately reveal just the payments—without the bids—to both the sellers and the winners, and prove to all bidders who did not win that their bid was not high enough to win. Thus the seller and every bidder are satisfied that the auction was conducted fairly, yet no information about the outcome of the auction needs to be published. Further transparency can be provided by requesting bidders "sign off" on their proven outcomes with a digital signature, so that the auctioneer can show that every bidder accepted the outcome. If a bidder refuses, the auctioneer can prove the outcome he provided was indeed correct by publicly revealing it.

6.4 Multi-Item Auctions

Consider now auctions for multiple identical items. In these auctions, the auctioneer has some number l of available identical items for sale. Real-life examples include large lots of refurbished items on eBay, or U.S. Treasury bills. We consider auctions in which bids are *flexible* and each bidder is willing to accept *any number of items up to a maximal limit and bid a price per item.* However, there is nothing about the framework that is limited in this way, and we will describe extensions to "all-or-nothing" bids and "bid curves" [137, 84] in future work.

As before, we can implement a general class of auctions that includes the first-price, uniform-price, and second-price (generalized Vickrey) auctions [85]. These are auctions in which the allocation depends only on the order of the bids and payments are defined as linear functions of the values of bids. For illustrative purposes we again

focus on the case in which the complete outcome of the auction, i.e. the allocation and all payments, is made public and then proved to be correct. Easy variants are available in which the correctness is selectively proved, either publicly for some restricted information about the outcome or privately to individual bidders.

6.4.1 Protocol

Step 1. *AU* posts the auction information on the bulletin board as in Section 6.3.1. In addition, *AU* posts the total number of items available, l, and the maximum allocation to any one bidder (if any), $l_{\max}$.

Step 2. Each participating bidder B_i prepares two integer values $(\mathbf{Bid}_i, \mathbf{Qty}_i)$ for each bid she wishes to submit to the auction, where $\mathbf{Bid}_i$ is the amount that she will pay per item and $\mathbf{Qty}_i$ is the maximum number of items desired by B_i.

As above, B_i also generates a random bit string σ_i and sends it to *AU*. B_i then encrypts $\mathbf{Bid}_i$ and $\mathbf{Qty}_i$, using *AU*'s public Paillier key n, as $E(\mathbf{Bid}_i)$ and $E(\mathbf{Qty}_i)$ and commits by sending *AU* and her notaries, if used, the commitment

$$\mathbf{Com}_i = [E_{TPK}(E(\mathbf{Bid}_i)), E_{TPK}(E(\mathbf{Qty}_i)), E_{TPK}(\sigma_i), ID], \tag{6.4}$$

and digital signature $\mathbf{Sign}_i(\mathbf{Com}_i)$. *AU* issues a receipt for these commitments and publishes them on the bulletin board in accordance with our standard protocol.

Step 3. As above, at time T, the auctioneer *AU* posts received commitments, his random string σ_{AU}, and test sets on the bulletin board. The number of test sets will depend on the type of the auction and the payment calculation; these numbers are detailed in Section 6.5.

Step 4. As above, bidders have between time T and $T+1$ to appeal non-inclusion, which may involve resorting to the commitments sent to any notaries.

Step 5. As above, bidders' encrypted bids and quantities $E(\mathbf{Bid}_i)$ and $E(\mathbf{Qty}_i)$, as well as their strings σ_i, are revealed between time T and $T+1$. *AU* publishes these values on the bulletin board. All bidders can check that the revealed values correspond with earlier commitments.

Step 6. *AU* privately recovers bids $\mathbf{Bid}_1, \ldots, \mathbf{Bid}_k$ and quantities $\mathbf{Qty}_1, \ldots, \mathbf{Qty}_k$ using secret key ϕ, and uses the information to compute the correct outcome of the auction. We again assume, without loss of generality, that the prices bid by bidders $B_1, \ldots, B_k$ are monotonically decreasing, though consecutive bids may be tied. We then choose the *threshold bid index*, α, which is new in our multi-item setting, such that bidders $\alpha, \ldots, B_k$ do not receive any items. The sum of the quantities associated with winning bids $\mathbf{Bid}_1, \ldots, \mathbf{Bid}_{\alpha-1}$ is greater than or equal to the number of available items l, and this is not true for a smaller threshold index. Thus all bidders B_i, such that $i < \alpha$, are winners. The threshold winner $\alpha - 1$ may receive some subset of her total demand. Formally, threshold index α is defined so that:

$$\left[\sum_{i=1}^{\alpha-2} \mathbf{Qty}_i < l\right] \wedge \left[\sum_{i=1}^{\alpha-1} \mathbf{Qty}_i \geq l\right] \tag{6.5}$$

Note that we have assumed here that there are enough bidders to cover all of the supply. This can be handled without loss of generality, by also introducing a single dummy bid at zero price for all supply, l. In addition to determining α, and thus the winners in the auction, *AU* also posts proofs of which bidders won and their allocations on the bulletin board, as well as proofs of the validity of each bidder's bid and quantity. He also computes proofs of correctness of each winner B_i's payment. If

public verification of payments is required, *AU* posts these correctness proofs on the bulletin board, along with the random help values needed to decrypt the payments. If the payments are to remain secret, he privately sends the proof for B_i's payment and any associated random help values to each winner B_i.

6.4.2 Verification

The verification step in a multi-item auction is more complex than for the single item auction, but relies largely on the same cryptographic primitives used in the simpler single-item case. Each verification can be done in a zero knowledge fashion, revealing no information beyond that implied by the outcome of the auction.

As before, *AU* first publicly proves the minimum *bid-ordering information*, that all winning bids are strictly greater than the threshold bid $\mathbf{Bid}_\alpha$, i.e., $\mathbf{Bid}_i > \mathbf{Bid}_{\alpha-1}$ for all $i < \alpha - 1$ and $\mathbf{Bid}_{\alpha-1} > \mathbf{Bid}_j$ for all $j \geq \alpha$. This reveals only minimum public information about the value of the bids; the same information that is implied by the outcome. *AU* will also prove that the bid values are valid and without wraparound. (See Section 3.2.3 for an explanation of wraparound.)

In addition, *AU* must also prove that the *quantities* of the items were encrypted correctly, i.e., without wraparound. We assume that $l < 2^t < n/2$ for number of available items l and test set size parameter t. *AU* first proves that no bidder has submitted a quantity greater than a specified maximum allowed allocation $l_{\max} \leq l$. To do this, *AU* first encrypts $E(l, 1)$ and $E(l_{\max}, 1)$; a help value 1 is used so that anyone can verify those encryptions. *AU* then proves $E(\mathbf{Qty}_i) \trianglelefteq E(l_{\max}, 1)$ for all $1 \leq i \leq k$. Next, *AU* can use encryptions of various sums of quantities to prove the

correctness of the threshold bid index α. Paillier's homomorphic encryption system allows for a zero-knowledge proof that a ciphertext represents the encrypted value of the sum of two encrypted values; in particular, $\prod_{i=1}^{\alpha-2} E(\mathbf{Qty}_i) = E(\sum_{i=1}^{\alpha-2} \mathbf{Qty}_i)$. Given this, *AU* can establish Eq. 6.5 over the encrypted quantities:

$$\left[E(\sum_{i=1}^{\alpha-2} \mathbf{Qty}_i) \lhd E(l)\right] \wedge \left[E(\sum_{i=1}^{\alpha-1} \mathbf{Qty}_i) \trianglerighteq E(l)\right] \tag{6.6}$$

Tiebreaking

In the event of a tie, with multiple bids equal in value to $\mathbf{Bid}_{\alpha-1}$, the auctioneer must also prove equality of these bid values and then establish correctness in allocating to these tied threshold bidders. Various algorithms exist for allocating the items among winners with equal bids at the threshold. One possibility is to randomly order the threshold bidders and divide the items among them in "round robin" fashion until the items are exhausted, with the condition that no bidder B_i is entitled to more than the $\mathbf{Qty}_i$ items bid for.

In this case, we require additional proofs that the allocation is fair. In summary, we use the random data σ jointly constructed by all auction participants to define a publicly verifiable ordering π of w equal bidders,[8] $\pi(1 \ldots w) \in \{1 \ldots k\}$ such that $B_{\pi(1)}$ is the first to be allocated an item, and so forth, and prove the round robin allocation as follows. We notate l_i as the allocation to bidder B_i.

Step 1. Prove that the allocations to all bidders add to l, i.e. $\sum_{i=1}^{k} l_i = l$.

Step 2. Given ordering π of threshold bidders, compute j such that $B_{\pi(j)}$ is the first bidder in the ordering to receive a partial allocation. Compute h such that $B_{\pi(h)}$

[8] Generating such a random ordering is described in Section 6.2.4.

is the first bidder in the ordering to receive $l_{\pi(j)} - 1$ items, i.e. the next bidder in line when the items ran out. If no such h exists, set $h = w + 1$.

Step 3. Prove that all allocations were fair as follows: *3a.* For $1 \leq i < j$, prove $l_{\pi(i)} = \mathbf{Qty}_{\pi(i)}$ and $l_{\pi(i)} < l_{\pi(j)}$. *3b.* For $j < i < h$, prove either that $l_{\pi(i)} = l_{\pi(j)}$, or both $l_{\pi(i)} = \mathbf{Qty}_{\pi(i)}$ and $l_{\pi(i)} < l_{\pi(j)}$. *3c.* For $h \leq i \leq w$, prove that $l_{\pi(i)} = (l_{\pi(j)} - 1)$, or both $l_{\pi(i)} = \mathbf{Qty}_{\pi(i)}$ and $l_{\pi(i)} < l_{\pi(j)}$.

In words, we show that bidders either received their entire allocation or at most one fewer than the first bidder in line to receive a partial allocation, and that the ordering of the partial allocations is proper.

Payment

In a *first-price* auction, the auctioneer can prove a payment to a third party by revealing the random help value used to encrypt winner B_1's bid. A verifier can use this to recover $\mathbf{Bid}_1$ from the now public encrypted value $E(\mathbf{Bid}_1)$ submitted by the bidder. Similarly, in a *uniform-price* auction, whereby every bidder pays the bid price of the losing threshold bidder $B_{\alpha-1}$, *AU* can provide a public proof by revealing $\mathbf{Bid}_{\alpha-1}$ via the help value used by $B_{\alpha-1}$. The uniform price auction is an approximation to a Vickrey auction in this setting. It generates the same payment as in the Vickrey auction to winning bidders $i < \alpha - 1$, as long as the threshold bidder has enough spare demand to cover the allocated capacity of any winner. The payment by the threshold winner $B_{\alpha-1}$ is always larger than in the Vickrey scheme.

We turn our attention to proving the correctness of prices in a generalized Vickrey auction (GVA) for this multi-item setting [85]. As in the single item setting, the GVA

provides the useful property of truthfulness so that each bidder's dominant strategy is to bid her true value per unit and true quantity demanded. In a GVA mechanism the number of items are allocated according to the price bid but the actual payment for each winner depends on others' bids. The Vickrey payment for bidder B_i is defined as:

$$p_{\text{vcg},i} = \mathbf{Qty}_i^* \cdot \mathbf{Bid}_i - [V(B) - V(B_{-i})], \tag{6.7}$$

where $V(B)$ is the total revenue in the auction with all bidders, $V(B_{-i})$ is the total revenue in the marginal economy with bidder B_i removed, and $\mathbf{Qty}_i^*$ denotes the quantity allocated to bidder i in the auction. This has a simple interpretation: a bidder's payment is determined as *the greatest amount other (displaced) bidders would have paid for the same items had B_i not been participating in the auction.*

We require a proof to establish the correctness of this payment. Let $\mathbf{Qty}_j^{-i}$ denote the quantity awarded to bidder B_j in the marginal auction without bidder B_i. For a non-marginal winner, i.e., $i < \alpha - 1$, her GVA payment is:

$$\begin{aligned}
&\mathbf{Qty}_i^* \cdot \mathbf{Bid}_i - \left[\mathbf{Qty}_i^* \cdot \mathbf{Bid}_i + \sum_{j\neq i, j\leq \alpha-1} \mathbf{Qty}_j^* \cdot \mathbf{Bid}_j\right] + \sum_{j\neq i, j\leq \beta_{-i}-1} \mathbf{Qty}_j^{-i} \cdot \mathbf{Bid}_j \\
&= \left[\sum_{\alpha-1<j\leq\beta_{-i}-1} \mathbf{Qty}_j^{-i} \cdot \mathbf{Bid}_j\right] + [\mathbf{Qty}_{\alpha-1}^{-i} \cdot \mathbf{Bid}_{\alpha-1} - \mathbf{Qty}_{\alpha-1}^* \cdot \mathbf{Bid}_{\alpha-1}]
\end{aligned} \tag{6.8}$$

For the marginal winner, $i = \alpha - 1$, her GVA payment is:

$$\begin{aligned}
&\mathbf{Qty}_i^* \cdot \mathbf{Bid}_i - [\mathbf{Qty}_i^* \cdot \mathbf{Bid}_i + \sum_{j\neq i, j<\alpha-1} \mathbf{Qty}_j^* \cdot \mathbf{Bid}_j] \quad + \sum_{j\neq i, j\leq \beta_{-i}-1} \mathbf{Qty}_j^{-i} \cdot \mathbf{Bid}_j \\
&= \sum_{\alpha-1<j\leq\beta_{-i}-1} \mathbf{Qty}_j^{-i} \cdot \mathbf{Bid}_j
\end{aligned} \tag{6.9}$$

Thus, the GVA payment by bidder B_i is a linear combination of the product of the bid price and allocated quantity to bidders displaced by bidder B_i from the winning allocation. In the case of a non-marginal bidder, this computation also accounts for the effect on the allocation to bidder $\alpha - 1$.

Consider the following verifiable proof structure for the term $\sum_{\alpha-1<j\leq\beta_{-i}-1} \mathbf{Qty}_j^{-i} \cdot \mathbf{Bid}_j$ that is common to both kinds of winners:

Step 1. In generating the proof, *AU* must first establish a bid ordering for the marginal auction without B_i, i.e., prove that β_{-i} is the correct threshold bid index by showing $\mathbf{Bid}_j > \mathbf{Bid}_{\beta_{-i}-1}$ for $j \neq i, j < \beta_{-i} - 1$ and $\mathbf{Bid}_{\beta_{-i}-1} > \mathbf{Bid}_j$ for $j \geq \beta_{-i}$; this can be done as in the main auction. Second, *AU* must prove that bidder $\beta_{-i} - 1$ is the threshold winner in this auction, by proving the analog to Eq. 6.5. Third, *AU* must publish encrypted values $\mathbf{Pay}_j = \mathbf{Qty}_j \cdot \mathbf{Bid}_j$ for all $j > \alpha_i, j < \beta_{-i} - 1$ (and similarly for the new marginal bidder, $\mathbf{Pay}_{\beta_{-i}-1} = \mathbf{Qty}_{\beta_{-i}-1}^{-i} \cdot \mathbf{Bid}_{\beta_{-i}-1}$), and prove the correctness of all of these ciphertexts. This requires proofs of *correct multiplication*, as described in Appendix 3.1. The proof of $\mathbf{Pay}_{\beta_{-i}-1}$ in turn requires a proof of the quantity allocated $\mathbf{Qty}_{\beta_{-i}-1}^{-i}$ to this bidder, via a proof that a published ciphertext is the encrypted value of $l - \sum_{j\neq i, j<\beta_{-i}-1} \mathbf{Qty}_j$. Fourth, *AU* must publish the encrypted value of the sum of these payments and a proof of its correctness.

Step 2. A verifier $\mathcal{V}$ can independently compute the encrypted Vickrey payment as above and check the correctness of the proof.

Step 3. *AU* reveals the random help value in the resulting encrypted Vickrey payment to $\mathcal{V}$, who decrypts using that value and verifies it is correct by re-encryption.

The verifier $\mathcal{V}$ now knows that B_i's Vickrey payment is correct while knowing

(almost) nothing more about any bidder's bid value than can be derived from the definition of Vickrey payments. In fact, the verifier $\mathcal{V}$ learns the *number* of bids required to compute a Vickrey payment in the marginal economy $\mathbf{E}(B_{-i})$. We can get around this through padding the input using dummy bids as described in the next section.

The additional term, $[\mathbf{Qty}^{-i}_{\alpha-1} \cdot \mathbf{Bid}_{\alpha-1} - \mathbf{Qty}^{*}_{\alpha-1} \cdot \mathbf{Bid}_{\alpha-1}]$ can be determined in the case that bidder i is the threshold winner and $i = \alpha - 1$ in an analogous fashion. Encrypted values of the allocation quantities received by bidder i in the main auction and in the marginal auction, i.e., $\mathbf{Qty}^{*}_{\alpha-1}$ and $\mathbf{Qty}^{-i}_{\alpha-1}$, can be established via subtraction from total items l of the total allocation to other bidders. Then, a ciphertext for the difference, $\mathbf{Qty}^{-i}_{\alpha-1} - \mathbf{Qty}^{*}_{\alpha-1}$, and then the product $(\mathbf{Qty}^{-i}_{\alpha-1} - \mathbf{Qty}^{*}_{\alpha-1})\mathbf{Bid}_{\alpha-1}$ can be published and proved.

Secrecy-Preserving Payment Proofs

While our above methods are correct, secure, and efficient in practice, they reveal a slight amount of additional information than that implied solely by the GVA payments. In particular, the method described to prove a GVA payment reveals the number of bidders whose bids in the marginal economy determine a bidder's price. This section outlines a more involved solution that eliminates the revelation of that information at some increased cost in complexity and computation.

We recall that the GVA payment for bidder B_i is defined as:

$$p_{\text{vcg},i} = \mathbf{Qty}^{*}_{i} \cdot \mathbf{Bid}_{i} - [V(B) - V(B_{-i})], \tag{6.10}$$

where $V(B)$ is the total revenue in the auction with all bidders, $V(B_{-i})$ as the total

revenue in the marginal economy with bidder B_i removed, and $\mathbf{Qty}_i^*$ denotes the quantity allocated to bidder i in the auction.

In order to prove the correctness of term $[V(B) - V(B_{-i})]$ we currently determine the threshold bidder β_{-i} in the marginal economy $(B \setminus i)$. Recall that the threshold bidder β_{-i} is defined so that all bids $< \beta_{-i} - 1$ receive a full allocation, bid $\beta_{-i} - 1$ may receive a partial allocation, and bids $\geq \beta_{-i}$ receive no allocation. But establishing the index of threshold bidder β_{-i} reveals information beyond that implied either by knowledge of the outcome of the auction or by the amount of an agent's GVA payment, specifically information about the number of bidders that were displaced by the presence of bid B_i.

To solve the problem we introduce a technique to prove the correctness of an encrypted term $V(B_{-i})$ without revealing any information about the number of winners in that marginal economy. This term can be used in combination with a proof of the correctness of term $V(B)$ and $\mathbf{Qty}_i^* \cdot \mathbf{Bid}_i$ to prove correctness for the GVA payment to bidder i.

To illustrate the idea we consider the case of proving correctness of the encrypted value of $V(B)$ for the main economy without revealing the index of the threshold bidder. Note also that a dummy bidder is included with bid 0 and quantity demanded l (the supply of items) when the total demand is less than l. Let k denote the total number of bids in the input, including this dummy bidder when required.

In order to hide the true index of the threshold bidder the idea is to pad the input with an additional $k - 1$ bids such that threshold index α given the padded input *is always defined so that* $\alpha - 1 = k$. Let γ denote the threshold index given the original

input of k bids. The new bids are defined as follows: there are $k-\gamma+1$ bids defined with $\mathbf{Qty}_j = 0$ and $\mathbf{Bid}_j = V$ for a maximal value V (higher than any posted bid), and $\gamma - 2$ bids defined with $\mathbf{Qty}_j = 0$ and $\mathbf{Bid}_j = 0$.

For example, if $k = 5$ then when $\gamma = 2$ (and only the first bid receives an allocation) then all $k-1$ new bids have $\mathbf{Qty}_j = 0$ and $\mathbf{Bid}_j = V$. On the other hand, when $\gamma = 6$ (and all bids receive some allocation) then all $k - 1$ new bids have $\mathbf{Qty}_j = 0$ and $\mathbf{Bid}_j = 0$.

Lemma. The threshold index of the padded input is equal to $k + 1$ and no information is learned about the threshold index in the initial index.

Moreover, the introduction of this padded input does not change $V(B)$ because the new padded bids demand no quantity and thus contribute nothing to the revenue of the auctioneer.

One problem remains with this solution: how do we ensure that the auctioneer can be trusted to introduce dummy bids with this property without revealing to the verifier the mixture of high value and zero value bids introduced? The verifier must not be able to tell whether a bid in the padded input is a dummy bid or an original bid, but still be confident that the auctioneer has provided a set of bids that contains exactly the posted bids and quantities along with correct padding.

For this we can again use the idea of "cut and choose":

Step 1. The prover constructs $2v$ test sets $TS_1, \ldots, TS_{2v}$. Each test set contains several bid collections.[9] TS_i contains k collections of $2k - 1$ bids, one collection for each of $m \in \{0, \ldots, k-1\}$ where there are m high value bids with quantity zero, the

[9]For clarity, we use the two words "collection" and "set", though there is no technical meaning differentiating the two terms.

k original bids and quantities, and $k-1-m$ low value bids with quantity zero. Each element of the collection is encrypted using the semantically secure Paillier scheme used elsewhere. Instead of encrypting the original plaintext bids and quantities, the auctioneer uses a re-encryption procedure (see Section 6.2.4) to yield an encryption of the same value that cannot be identified as such. The $2k-1$ elements of each collection are permuted randomly; the k collections within each test set are also permuted randomly. Now, each test set contains k collections of $2k-1$ bids, where each collection is the original auction's bids padded with dummy bids and zero quantities. These test sets are posted.

Step 2. The verifier randomly selects v test sets, requests that the prover identify each of the elements in every collection as either a high value, low value, or posted bid, and prove that fact by revealing the random help values (for dummy bids and their quantities) or the random re-encryption factor s and the original bid or quantity (for re-encrypted posted values). If there is a problem then the procedure is aborted and the verifier requests a new list of $2v$ fresh test sets.

Step 3. There remain v unexamined test sets. The prover will use each of these to construct a proof as follows: for each test set, the prover identifies one of the collections of bids within that test set, and then completes the proof of the value for $V(B)$ using the padded input with that collection of bids. Not only must the payment be the same for each padded input but the threshold index, given the padded input, must always be $k+1$. As before the value v may be selected to provide a desired probability of error in the outcome.

Because we do not want to reveal which bid is in position α given marginal

economy $(B \setminus i)$ and thus do not compute (and prove) the total revenue from bids $\{\alpha, \ldots, \alpha_{-i} - 1\}$ we prove instead the total value of $V(B_{-i})$ in this new approach rather than establishing directly the loss in revenue as a result of bid $\mathbf{Bid}_i$ directly, as in the previous section.

This approach can also be used to prove to each bidder the correctness of her allocation without revealing the number of winners, and similarly to prove to any third party the correctness of any single bidder's allocation. As described in Section 6.3.3, this may have special importance in auctions in which it is desirable for partial information about the outcome to be privately proven to some parties; for example, it may be desirable for the outcome to be secret while the seller, each bidder, and perhaps a third-party auditor still receive a proof that the auction outcome is correct.

6.4.3 Extensions

We assume each bidder submits only one bid/quantity pair, but a single bidder could simply submit multiple bids in order to represent a more complex utility function. The auction will have the correct behavior when used with first-price or uniform-price payment schemes. For example, a bidder might wish to purchase 10 units if the price is \$50, but 30 units if the price is \$40. By placing two bids, $(\$50, 10), (\$40, 20)$, the bidder will receive, for example, 30 units if the threshold for winning bids is less than \$40, 10 units if the threshold is between \$40 and \$50. While this "additive-or" bidding logic does not permit bidders to specify completely arbitrary utility functions, it does provide additional expressivity. Note, though, that if this language is used in an auction with GVA payments the bidder's payment could

be too high. The logic of GVA requires removing *both* of its ($50,10) and ($40,20) bids when computing its payment, but this would not automatically happen when considering these as separate bids. Extensions to correctly handle GVA payments with more expressive languages [112], as well as methods to adopt more expressive languages in which bidders can submit a set of bids with explicit logical dependencies, are reserved for future work.

6.5 Empirical Results

We implemented Paillier encryption and test set verification in C++ using the LiDIA number theory package [93] on a commodity Linux workstation with a Pentium 4 2.8 GHz processor.

The greatest computational cost in our protocol is the construction and verification of test sets, and in particular the exponentiation of random help values (r^n) required to encrypt or (verifiably) decrypt a value. These calculations dominate all other computation; for example, to sort one million random 64-bit bids takes less than one second on our system. In a single-item auction, the auctioneer can prepare for an auction of 100 bidders in about two hours, and each verifier can independently verify the auctioneer's proofs of correctness in less than half an hour. Both preparation and verification scale linearly and are easily parallelized. Thus, with modest distributed computation, even a multi-item auction with ten thousand bidders can be prepared in a few hours and verified in reasonable time.

We present data for both 1024- and 2048-bit symmetric public encryption keys, which are considered safe until 2010 and 2030, respectively [69]. Because the lifetime

of a security key is based on the difficulty of breaking it on available computing power, we claim that, for the most part, an auction with "5-year" security at any point in time will take at most about the same amount of time as it does today, as improvements in computing power for breaking keys are likely to be comparable to those in encryption.[10]

Table 6.1 shows the time it takes to compute various cryptographic operations on our test machine. We observe that the time required to prepare or verify a test set is essentially that required by the encryption and decryption. All test sets represent 2^{34} discrete values.

For a single item auction of k bidders, the auctioneer must produce k proofs of valid bids (i.e. $\mathbf{Bid}_i < 2^t$ for small t; we use 34), and $k-1$ proofs of comparisons to prove the ordering of the outcome. Using the bulk verification method suggested in Appendix 3.2.3, such an auction requires $10 \cdot (2k-1)$ test sets, plus 25% for the test sets that will be revealed to prove the test sets are valid. This gives us an upper bound of $25k$ test sets required to conduct a trustworthy single-item auction.

For a multi-item auction with payments based on one bid (e.g. first-price or second-price), we need only add to the above k proofs $\mathbf{Qty}_i < 2^t$, k comparisons $\mathbf{Qty}_i < l_{\max}$, and 2 comparisons to prove Equation 6.5. This means we need about double the number of test sets, $4k+1$, to conduct such a multi-item auction; about $50k$ test sets are needed for trustworthiness. We list the time taken to prepare these test sets and correctness proofs in Table 2.

For verified GVA payments in multi-item auctions (Section 6.4.2), we also require

[10] Of course, if the cryptographic assumptions underlying our protocols are discovered to be untrue, our claim does not hold.

proofs of multiplications for at most $2k + 1$ products, namely, $\leq k$ proofs of the products $\mathbf{Qty}_i \cdot \mathbf{Bid}_i$ and $k + 1$ proofs of the products of the partial allocation to the threshold bidder for the main economy $\mathbf{E}(B)$ and up to k marginal economies (that is, excluding bidder B_i) $\mathbf{E}(B_{-i})$. Each proof of a product requires 4 exponentiations for creating the MTS ("multiplication test set") and 6 exponentiations to verify it. To achieve a reasonably small probability of error, we need to repeat the multiplication proof 80 times ($\frac{3}{4}^{80} \approx 10^{-10}$). Thus each proof requires 320 exponentiations to create and 480 to verify. Table 6.3 shows time required, again on a P4 2.8 GHz processor, to verify Vickrey payments in the worst case for various sizes of multi-item auctions. These computations are required *in addition* to the above computations for verifying prices and quantities.

6.6 Conclusions and Future Work

We have presented a new protocol for sealed-bid auctions that guarantees trust and preserves a high level of secrecy, yet is practical enough to run efficiently on commodity hardware and be accepted in the business community. Because we focus on proofs of correctness and secrecy during the auction, an auctioneer can still compute optimal results efficiently and publish efficiently verifiable proofs of those results. Our protocol rests on sound cryptographic foundations, and lends itself to interesting extensions to further types of auctions, including support for all-or-nothing bids, bid curves, and full combinatorial auctions; we intend to pursue these extensions in later work. We believe that our practical, easily implemented approach can be extended to other areas of privacy, including electronic transactions, trading systems, privacy-preserving open

outcry markets, and zero-knowledge public verification of private data. Along these lines, authors Thorpe and Parkes have recently extended our methods to a continuous double auction setting for information hiding in securities exchanges [150].

To further explore the practicality of our solution, David Austin has built a prototype of our protocol. His Python implementation comprises a fully functional, cross-platform web server and standalone client for creating, bidding on, and verifying sealed-bid auctions. Because it is implemented in Python, it runs at approximately half the speed of our empirical tests, which were conducted using optimized C++, but is still fast enough for practical use.

Other future work includes improving the efficiency of our protocols. Due to the dominance of range proofs in auctions, employing more efficient techniques to prove an encrypted value in a particular range are likely to reduce the computation required to prove an auction correct (see Section 3.2.3, and [36, 44, 30, 79, 128] cited there). Use of specialized cryptographic hardware for performing modular exponentiation of very large integers instead of standard 32- and 64-bit hardware may also yield significant time savings. Finally, it may be that for many auctions, the auction data need only be secure *during* the auction, and not for years later, and thus shorter cryptographic keys might be employed at a significant savings in computational cost.

While this chapter focuses on auctions in which price is the only consideration, non-price factors such as technical quality, terms of payment, and service agreements, are of course also important in auctions used for procurement. However the effect can be to make the rules of the auction "soft" and provide new opportunities for corruption, since the auctioneer has new flexibility to manipulate the outcome of the

auction in return for a bribe [134, 38].

Of course, the use of cryptographic methods to verify the correct outcome of an auction requires *objective* criteria for determining the outcome based on the bids. It is useful, then, that concerns about corruption have led the World Bank and other bodies to move towards requiring *quantifiable decision making*, with the revelant "scoring" criteria published as part of the rules of the auction [151, 13, 149].

This makes quality assessment objective and reduces the corruption concerns to those of bid rigging in price-based sealed-bid auctions. As such, it is of significant interest in future work to develop provably correct, and trustworthy auctions by appropriate extensions to our technology. We also plan to study the use of similar technology in cryptographic *open-bid* settings, beyond the combinatorial clock [124] auctions described in Chapter 7.

Table 6.1: Time to perform basic operations

Operation	**Time** (s.) (1024-bit)	**Time** (s.) (2048-bit)
Computation of r^n	0.045	0.287
Encryption	0.045	0.287
Decryption with r	0.045	0.287
Decryption with ϕ	0.014	0.089
Decryption with r^n	0.000	0.001
Constructing a TS	3.01	19.32
Verifying a TS	3.00	19.30
Proving $0 \leq x < 2^t$ given TS	0.001	0.001
Verifying proof of $0 \leq x < 2^t$	0.070	0.41

Table 6.2: Time to prepare and verify auctions

Operation	**Number of Bids**		
	100	1000	10000
	Single-item Auctions		
Preparation (1024-bit)	2.1 hr	21 hr	8.7 days
Verification (1024-bit)	25 min	4.2 hr	42 hr
Preparation (2048-bit)	13.4 hr	5.6 days	56 days
Verification (2048-bit)	2.7 hr	27 hr	11 days
	Multi-item Auctions		
Preparation (1024-bit)	4.2 hr	42 hr	17.5 days
Verification (1024-bit)	52 min	8.7 hr	3.6 days
Preparation (2048-bit)	27 hr	11.2 days	112 days
Verification (2048-bit)	5.4 hr	54 hr	22 days

Table 6.3: Verification of Vickrey payments for multi-item auctions

Operation	**Number of Bids**		
	100	1000	10000
Preparation (1024-bit)	48 min	8 hr	3.3 days
Verification (1024-bit)	72 min	12 hr	5 days
Preparation (2048-bit)	5.1 hr	51 hr	21 days
Verification (2048-bit)	7.7 hr	77 hr	32 days

Chapter 7

Cryptographic Combinatorial Clock Proxy Auctions

We present a practical cryptographic protocol for conducting efficient, provably fair and secrecy-preserving combinatorial clock-proxy auctions. During the clock phase, bidders submit encrypted bids and prove for themselves that they meet activity rules. Bidders can also compute the total demand without revealing any information about individual demands. The effect is to make the clock-proxy auction function as a trusted sealed-bid, proxy auction despite the price discovery phase. Once the auction closes, all bids are revealed to the auctioneer who can then employ efficient branch-and-bound algorithms to determine the outcome of the proxy auction. We demonstrate the use of homomorphic encryption to prove the correctness of solutions to problems of mathematical optimization. The outcome of the clock-proxy auction must be accompanied by a cryptographic proof that is published by the auctioneer and establishes the correctness of solutions to optimization problems by reasoning

about properties of linear optimization. Any party can verify the correctness of the outcome and the fairness of the complete process.

7.1 Introduction

While there now exist practical protocols for cryptographic auctions of identical items, and practical methods of computing optimal outcomes in non-cryptographic combinatorial auctions, we know of no practical protocol for conducting a *cryptographic combinatorial auction*, in which a seller offers various quantities of distinct goods, buyers bid on bundles of these goods, and cryptography provides both secrecy and provable correctness. By secrecy, we mean that the auctioneer cannot exploit bid information to change the outcome of the auction, and by provable correctness, we mean that the auctioneer is obligated to issue proofs of correctness to prove he did not deviate from the posted auction rules.

Indeed, the optimization problem associated with combinatorial auctions is NP-complete; computing the outcome of such an auction in a secure manner is therefore a significant challenge. We describe a cryptographic auction protocol that it meets our secrecy and provable correctness requirements, elicits accurate bids, and can be implemented in a realistic business setting on cost-effective computing hardware. As an important component of this protocol, we develop a general framework for proving the correctness of a solution to mathematical optimization problems where the input and constraints are encrypted.

The particular combinatorial auction that we study is the *combinatorial clock-proxy auction* (CCP) [15], a simple and efficient protocol for conducting combinatorial

auctions. It was originally developed for auctions of wireless spectra but is applicable to other complex domains. Its principal merits are that it combines a simple price discovery ("clock") phase with a last-and-final round implemented as a sealed-bid combinatorial ("proxy") auction.[1]

In the clock phase, the auctioneer creates a "clock" for each item for sale that represents the current price at which that item is to be sold, starting with low prices. In a sequence of similar rounds, bidders submit a bundle of the items they desire at the current clock prices. Whenever the demand exceeds the supply for a good, the clock price increases for that good in the next round. The clock auction ends when there is no excess demand for any good. At this point bidders can submit additional bids, which, together with the clock bids, form the bids that define the input to the proxy auction. The proxy auction is a generalized second price, sealed-bid auction that reduces to the Vickrey-Clarke-Groves (VCG) mechanism in special cases while avoiding some of VCG's undesirable economic properties in other cases [16], including problems related to low revenue and vulnerability to collusion by losing bidders.

In our *cryptographic combinatorial clock proxy* (CCCP) auction, all bid information is encrypted, and these encryptions are posted to the public. No party, including the auctioneer, can decrypt any values until all bids have been submitted in both phases. After all bids are in, only the auctioneer receives the decryption key, computes the outcome in private, reveals individual outcomes to each bidder, and issues efficiently checkable proofs that the reported outcomes are correct given the public encrypted bids. This complete secrecy until the auction closes removes opportunities

[1] Porter et al.[124] earlier described a combinatorial-clock auction and Parkes and Ungar [120, 121] and Ausubel and Milgrom [16] earlier described variants on the proxy auction.

for collusion while assuring that the process remains trusted and verifiable by all participants, offering an unprecedented balance of efficiency, privacy, and transparency.

In non-cryptographic auctions, trust is made possible at the cost of privacy: disclosure. Indeed, this is one path that Ausubel et al. [15] suggest. But this can be undesirable for a number of reasons: bidders may not want competitors to learn about the values of their bids even after the fact; it may be politically undesirable to reveal that the winning bidder was willing to pay much more that was charged via the auction rules, and revealing bids received during the clock phase may lead to opportunities for collusion.[2] Ausubel et al. [15] also argue that the confidentiality of values is of primary importance in an implementation, and suggest that in some areas of the auction, some values should be hidden even from the auctioneer: "Only the computer need know." Our techniques complement such a "black box" system by guaranteeing the results are correct, not simply that the programs on the system are believed to be correct.

We advance several contributions in the present work. During the clock phase, we employ homomorphic cryptography to protect the secrecy of bids while allowing bidders to prove they satisfy activity rules and allowing everyone to compute the aggregate demand for goods that determines the next round's prices. As in our previous work on non-combinatorial sealed bid auctions [119], we employ time-lapse cryptography [129], to provide secrecy during the bidding process while enforcing *nonrepudiation*: guaranteed revelation of the bids to the auctioneer when the bidding

[2]In a recent FCC auction for the 700MHz spectrum the government has for the first time removed all feedback about the particular bids submitted in each round. Each bidder receives individualized feedback about its own bid activity. Clearly this higher degree of secrecy brings along the need for increased trust in the auctioneer.

is complete. This avoids protocol completion incentive problems [31] in which bidders who realizing they will lose or change their minds can refuse to complete a distributed commercial protocol.

In the primary technical contribution, we demonstrate how to use our cryptographic framework to prove the correctness of solutions to general classes of linear optimization problems; this is how we efficiently compute the auction outcome and prove it correct. Our auctioneer employs powerful branch-and-bound mixed-integer programming search techniques to compute the outcome in private, avoiding costly secure computation for the optimization task; he can then prove that the outcome is correct with efficiently checkable proofs. This allows us to support much larger-scale combinatorial auctions than in the current literature while maintaining the same level of provable fairness.

7.1.1 Related work

A body of existing research considers the use of cryptographic methods to provide trust without compromising privacy; see Brandt [33] and Parkes et al. [119] for a recent discussion. Much of the previous work focuses on non-combinatorial sealed bid auctions with *complete privacy*, where no party learns anything except the outcome [65, 72, 110, 94]. We advanced in [115, 119] the security model we adopt here, that of an auctioneer who must prove every action correct, and who learns bid information only after the auction closes—preventing meaningful disclosures.

We are only aware of one collection of research, by Yokoo and Suzuki, that considers cryptographic combinatorial auctions in depth; while their pioneering work offers

a theoretical solution to an important problem, their solutions, which require exponential computations to prove the auction correct, scale only to very small auctions in practice. One method they provide is based on dynamic programming using polynomial secret sharing to compute the optimal solution to the combinatorial optimization problem without revealing the inputs [146]. Another method that they describe employs homomorphic encryption [147], as is natural to adopt for our system, but again fails to scale because computation is performed explicitly on each of the exponentially many possible allocations of goods. The same authors later extended their work to remove the need for a third-party auctioneer [156], but are limited by the scalability of dynamic programming in this domain and also by additional process complexity implied by such a completely distributed solution. Finally, a related paper by Naor, Pinkas and Sumner [110] proposes the use of garbled circuits to compute the outcome of a combinatorial auction. Though the work is important for its foresight and theoretical affirmative results, we know of no practical implementation of obfuscated circuits that has been applied to significant real-world problems on the scale of a commercial combinatorial auction.

7.2 Cryptographic preliminaries

Several cryptographic systems support the secrecy-preserving, provably correct computation we employ to conduct the auction. Because Paillier's cryptosystem [114] supports all of the operations we employ and is widely accepted in secure protocols, we use it in our exposition of a complete system to conduct a CCCP auction. That said, there is nothing that necessitates the use of Paillier's system; in fact, other

solutions can be constructed that are computationally more efficient but complicate the protocol. These include, among others, Pedersen [122] commitments and ElGamal encryption [58], based on the hardness of computing discrete logarithms modulo a prime, and the provably correct secure computation system described by Rabin et al. [128].[3] We reserve for future work a complete discussion of how these and other systems might also support our protocol.

The Paillier-based implementation from Chapter 3 of our general framework described in Chapter 2 offers us a full set of provably correct, secrecy-preserving mathematical operations on encrypted inputs.

7.2.1 Mix Networks

Due to special mathematical properties Paillier encryption enjoys, it is possible for a Prover (in our application the Auctioneer) to create a random permutation S' of a set of encryptions S so that a verifier believes that S' encrypts precisely the same set of values that S does. In the spirit of our work, this can be done in a manner not revealing any information about the encrypted values.

In the Paillier cryptosystem, one can generate a new "random-looking" encryption of a particular element by multiplying it by a encryption of 0 — we call this a "re-encryption factor". The auctioneer can create many random permutations of the encrypted values and commit to the re-encryption factors in each permutation. The verifier then asks the auctioneer to reveal the re-encryption factors for some of the

[3]We have devised a similar protocol to the one we describe based on Pedersen commitments; while this protocol is computationally more efficient, it is mathematically more sophisticated, and we present the solution here because of the simplicity a protocol with a single cryptosystem enjoys.

permutations, and verifies that the factors are well-formed (that is, they are encryptions of zero) and that the permutation is correct. The remaining permutations, for which the factors remain unrevealed, are now verified correct with high probability.

Cryptographers have formalized this idea as a "shuffle", or "mix network".[4] See Abe et al. [4, 5] for early work on such permutation networks, and Boneh and Golle [28] for an excellent formalization of mix networks, a brief survey of other solutions, and an interesting efficient protocol for proving a mix network is correct with high (but not overwhelming) probability.[5] We will employ a mix network to create a verifiable random permutation of the encrypted bids that are submitted to the proxy auction. This will allow the branching decisions of the branch-and-bound proof tree to be published without revealing any information about the actual underlying inputs to the linear optimization problems; bidders can thereby be satisfied with their outcome without learning private bid information.

7.3 Combinatorial Auction Preliminaries

We consider a multi-unit combinatorial allocation problem with goods $G = \{G_1, \ldots, G_m\}$ and bidders $B = \{B_1, \ldots, B_n\}$. There are C_j units of each good G_j available and each bidder B_i has a valuation function $v_i(s_i)$ on bundles $s_i \in \mathbb{Z}^m_{\geq 0}$, where $s_{ij} \leq C_j$ denotes the number of units of item G_j in the bundle. An efficient allocation solves $V^* = \max_{s \in \mathbb{F}} \sum_i v_i(s_i)$ where $\mathbb{F} = \{s : \sum_i s_{ij} \leq C_j,\ \forall j \in G\}$ and $s = (s_1, \ldots, s_n)$ denotes the allocation of items to bidders. We assume quasi-linear

[4]The latter term should not be confused with hard-to-trace network communications protocols that are sometimes referred to by the same name.

[5]Boneh and Golle's efficient solution should not be employed without using an additional mechanism to verify its correctness. See [28].

utility u_i (or *payoff* π_i), so that bidder B_i's utility for bundle s_i, given payment $y_i \in \mathbb{R}_{\geq 0}$, is $\pi_i = u_i(s_i, y_i) = v_i(s_i) - y_i$. We make the standard assumptions of *normalization*, with $v_i(s_i) = 0$ when $s_{ij} = 0$ for all items G_j, and *free disposal*, with $v_i(s_i) \geq v_i(s_i')$ for $s_i' \geq s_i$.

The payments in the proxy auction implement a particular outcome on the *buyer-optimal core*. Consider the payoff vector $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ induced by an efficient allocation s^* and payment vector $y = \langle y_1, \ldots, y_n \rangle$. Let π_0 denote the payoff to the seller, which is the total revenue received by the seller with $\pi_0 = \sum_i y_i = V^* - \sum_i \pi_i$. A payoff profile $\langle \pi_0, \pi \rangle$ is in the core if $\pi_0 + \sum_{i \in K} \pi_i \geq V(K)$ for all $K \subseteq B$, where $V(K) = \max_{s \in \mathbb{F}} \sum_{k \in K} v_k(s_k)$. This states that no coalition of $K \subseteq B$ buyers and the seller can improve its total payoff by leaving the auction and allocating the items amongst itself, leaving all members weakly better off. Simple algebra shows that the core payoffs can be equivalently defined as $Core =$

$$\{\pi : \sum_{i \in W \setminus K} \pi_i \leq V^* - V(K),\ \forall K \subseteq W, \pi_i \geq 0, \pi_i \leq v_i(s_i^*)\},$$

where W is the set of *winners* in the efficient allocation s^*.

The *buyer-optimal core* defines a payoff vector that solves $\overline{\pi} \in \arg\max_{\pi \in Core} \sum_i \pi_i$. We can relate the buyer-optimal core to the outcome of the Vickrey-Clarke-Groves (VCG) mechanism. The VCG mechanism defines payments so that the payoff to bidder i is $\pi_i^{\text{vcg}} = V^* - V(B \setminus \{i\})$, i.e., each bidder's payoff is the marginal value it contributes by its presence. The buyer-optimal core is unique and coincides with the VCG outcome when the VCG outcome is in the core. But in general we have $\sum_i \overline{\pi}_i < \sum_i \pi_i^{\text{vcg}}$ and the revenue to the seller is greater in a buyer-optimal core outcome than in the VCG mechanism. In the particular instantiation of the proxy auction

that we consider in this chapter, when the buyer-optimal core is non-unique, the final payments are computed to minimize the maximal difference to the VCG payoff across all buyer-optimal core outcomes. This particular choice follows the suggestion of *threshold payments* in Parkes et al. [118] in the context of a combinatorial exchange, and as refined in the context of the proxy auction by Day and Raghavan [52].

7.4 Conducting the Clock Auction

Our presentation of our main results begins by considering the first phase of CCCP, which is the *clock auction* phase. The clock phase proceeds in rounds until demand does not exceed supply for any good. In each round t, a price vector $p^t = \langle p_1^t, \ldots, p_m^t \rangle$ associates prices with each good: p_j^t is the price for good G_j in round t. The price vector is initialized to low prices (although not necessarily uniformly across all goods) for the first round, $t = 1$, and is increased in each successive round based on the amount of excess demand. Bidders submit a bid $s_i^t \in \mathbb{Z}_{geq0}^m$ in each round. These bids are ultimately included within the proxy bids that form the input to the proxy phase.

The main challenge that we face in the clock phase is to allow for the price discovery process but without allowing any party—the auctioneer included—to learn anything about any bids not already implied by the public information. Following the description of Ausubel et al. [15], we allow the price increase on a good in a round to depend on the amount of excess demand on that good.[6] One requirement, then, is that any party (the auctioneer included) must be able to determine the excess

[6] Ausubel et al. [15] also discuss the idea of using *intra-round bids* in which the auction proceeds in a smaller number of discrete rounds and bidders express quantity demands in each round at all prices along a price trajectory that will be traced during the round. We save this extension for future work.

demand on each good in the current round without learning anything else about the current bids. It will also be necessary to allow any party to verify that the bids meet a *revealed preference activity rule* (RPAR) without revealing any information.

All bids made during the clock phase must also be submitted as proxy bids in the proxy phase. We ensure this and prevent non-repudiation through the use of a time-lapse cryptography (TLC) service [129]. At the start of the auction, the auctioneer in CCCP announces the initial price vector p^1 and the supply $C = \langle C_1, \ldots, C_m \rangle$ and designates a *public time-lapse cryptographic key* N, as described in Section 6.2.4. Because the secret key corresponding to N (and based on the factorization of N) is not revealed until after all bidder information has been submitted, the auctioneer cannot reveal private information that could affect the outcome. The forced reconstruction of N guarantees that the bids can be opened by the auctioneer when the auction is complete.

7.4.1 A Sequence of Clock Rounds

At the beginning of round t, the auctioneer publishes the current clock price vector $p^t = \langle p_1^t, \ldots, p_m^t \rangle$. Then, each bidder B_i publishes an encrypted version of her bid given the current prices: $E(s_i^t) = \langle E(s_{i1}^t, r_{i1}^t), \ldots, E(s_{im}^t, r_{im}^t) \rangle$. Bidders publish these encrypted bundles to all bidders, the auctioneer and any verifiers, either by broadcast or to a common "bulletin board" during a fixed period of time for round t. This encrypted bundle is represented as a vector of length m, in which each coefficient s_{ij}^t is an encryption of the quantity B_i wants for good G_j at price p_j^t. The values r_{ij}^t are independent, fresh *random help values* that each bidder selects in accordance with the

probabilistic homomorphic encryption scheme, and kept secret. Encryptions of zero must be included for any undesired item to keep the number of items in the bundle secret.

Bid Validity and Activity Rules

Each bidder must now prove that the bid is valid and satisfies an *activity rule*.[7] The basic idea in a revealed-preference activity rule (RPAR) is to require bidders to follow a demand-revealing strategy that is consistent with some fixed valuation function across all clock rounds. Consider a current round t and some previous round $t' < t$, corresponding price vectors p^t and $p^{t'}$, and B_i's associated demands s_i^t and $s_i^{t'}$. A straightforward bidder with valuation v_i prefers s_i^t to $s_i^{t'}$ when prices are p^t:

$$v_i(s_i^t) - p^t \cdot s_i^t \geq v_i(s_i^{t'}) - p^t \cdot s_i^{t'}$$

and prefers $s_i^{t'}$ to s_i^t when prices are $p^{t'}$:

$$v_i(s_i^{t'}) - p^{t'} \cdot s_i^{t'} \geq v_i(s_i^t) - p^{t'} \cdot s_i^{t'}.$$

Adding these two inequalities (the values of the bundles cancel) yields the activity rule:

$$(p^t - p^{t'}) \cdot (s_i^t - s_i^{t'}) \leq 0.$$

Before proving the RPAR, bidders must prove that their current demands are valid by using an interval proof: each B_i proves for the demand for good G_j, $0 \leq s_{ij}^t \leq C_j$. That is, the demand lies in the interval between 0 and the auction's capacity for that good.[8]

[7] While we talk about the "bidder" proving various facts about the bid history to the auctioneer and any other interested party, we of course intend the proofs to be generated by a computer program running on secure hardware controlled by the bidder, both to maintain the security of any private information and because the cryptographic computations should not be carried out by hand.

[8] We also require that the capacities C_j are less than half the modulus of the cryptosystem ($N/2$),

Each bidder can now readily prove that she satisfies the activity rule using homomorphic cryptography via the clock prices and the published encrypted bids. This must be established in round t with respect to all previous rounds $t' < t$. First, since the price vectors $p^{t'}$ and p^t are public, anyone can compute the price difference vector $\hat{p} = \langle \hat{p}_1, \ldots, \hat{p}_m \rangle = p^t - p^{t'}$. Second, using the encrypted demand vectors $E(s_i^t)$ and $E(s_i^{t'})$, the homomorphic properties of the cryptosystem allow computing B_i's encrypted demand difference vector $\hat{s}_i = \langle \hat{s}_{i1}, \ldots, \hat{s}_{im} \rangle = s_i^t - s_i^{t'}$:

$$
\begin{aligned}
E(s_i^t) &= \langle E(s_{i1}^t, r_{i1}^t), \ldots, E(s_{im}^t, r_{im}^t) \rangle \\
E(s_i^{t'}) &= \langle E(s_{i1}^{t'}, r_{i1}^{t'}), \ldots, E(s_{im}^{t'}, r_{im}^{t'}) \rangle \\
E(\hat{s}_i) &= \langle \frac{E(s_{i1}^t, r_{i1}^t)}{E(s_{i1}^{t'}, r_{i1}^{t'})}, \ldots, \frac{E(s_{im}^t, r_{im}^t)}{E(s_{im}^{t'}, r_{im}^{t'})} \rangle \\
&= \langle E(s_{i1}^t - s_{i1}^{t'}, r_{i1}^t / r_{i1}^{t'}), \ldots, E(s_{im}^t - s_{im}^{t'}, r_{im}^t / r_{im}^{t'}) \rangle
\end{aligned}
$$

We then want to compute the encrypted dot product of the price difference vector and the encrypted demand difference vector, that is, $E(\hat{p} \cdot \hat{s}_i)$. Again using the homomorphic properties of the cryptosystem, we can compute this single coefficient as follows,

$$
\begin{aligned}
E(\hat{p} \cdot \hat{s}_i) &= E(\hat{s}_{i1}, r_{i1}^t / r_{i1'}^t)^{\hat{p}_1} \times \ldots \times E(\hat{s}_{im}, r_{im}^t / r_{im}^{t'})^{\hat{p}_m} \\
&= E(\hat{p}_1 \times \hat{s}_{i1}, r_{i1}^t / r_{i1}^{t'}) \times \ldots \times E(\hat{p}_m \times \hat{s}_{im}, r_{im}^t / r_{im}^{t'}) \\
&= E(\hat{p}_1 \times \hat{s}_{i1} + \ldots + \hat{p}_m \times \hat{s}_{im}, r_{i1}^t / r_{i1}^{t'} \times \ldots \times r_{im}^t / r_{im}^{t'})
\end{aligned}
$$

We adopt $\hat{r}_i$ to notate the random help value encrypting the dot product (the last formula above): $\hat{r}_i = r_{i1}^t / r_{i1}^{t'} \times \ldots \times r_{im}^t / r_{im}^{t'}$. We now have an encryption of this dot

but as the moduli are typically hundreds or thousands of bits, this poses no practical problems.

product—a single value that proves the activity rule when it is less than or equal to zero.[9] Consequently, each bidder now proves using another interval proof (see Section 3.2.3) that this encrypted value is less than (but relatively close to) zero. Our example shows that B_i can compute the precise random help value corresponding to the encryption of a dot product of an encrypted vector with a public vector. This allows B_i to prove facts about the result like any other value it encrypted and even though the decryption key has not yet been constructed.

Computing Aggregate Demand

At the conclusion of each round, the aggregate demand for each item must be computed. This is done in a similar way, using the homomorphic properties of the cryptosystem. The aggregate demand vector s^t for all goods at the end of round t is simply:

$$s^t = \langle \sum_{i=1}^{n} s_{i1}^t, \ldots, \sum_{i=1}^{n} s_{im}^t \rangle$$

Given the encrypted demand vectors, we can compute an encryption of the aggregate demand vector s^t as follows:

$$E(s^t) = \langle \prod_{i=1}^{n} E(s_{i1}^t, r_{i1}^t), \ldots, \prod_{i=1}^{n} E(s_{im}^t, r_{im}^t) \rangle \tag{7.1}$$

$$= \langle E(\sum_{i=1}^{n} s_{i1}^t, \prod_{i=1}^{n} r_{i1}^t), \ldots, E(\sum_{i=1}^{n} s_{im}^t, \prod_{i=1}^{n} r_{im}^t) \rangle \tag{7.2}$$

By multiplying each bidder's encrypted demand for an item together, we obtain an encryption of the sum of all bidders' demands for that item; the random help value of this encryption is the product of the random help values from all bidders'

[9] If the bidder does not prove the activity rule, then the bid is invalid and the auction rules should dictate whether the bidder must resubmit or is disqualified for the round.

encrypted demands. Since the secret decryption key does not yet exist, decryption can only be performed by unlocking the encrypted value with its random help value.

While the random help value could be directly constructed from the other values, such a direct computation would reveal too much, because each encrypted demand's random help value would unlock that particular demand. We thus employ another well-known cryptographic protocol to compute the random help values needed to unlock the aggregate demand for each good, which we detail in Section 3.3. This process is repeated after each round t, for each good G_j, to compute the above aggregate demand vector (Eq. 7.2). B_i constructs shares of the random help value associated with the demand for good G_j, so that the product of these shares equals the random help value r_{ij}^t. B_i then distributes these shares among all bidders. Once all the shares are received, the bidders multiply their received shares together, yielding random factors of the help value $\prod i = 1^n r_{ij}^t$. Then, bidders broadcast these random factors to all bidders, and multiply them together to yield the desired help value. This allows anyone to decrypt the encrypted sum of the aggregate demand for that good and verify the result. Recall that since the encrypted individual demands are public, one can compute an encryption of their sum by multiplying the encryptions.

We remark without proof that this sub-protocol to compute the random help values is information-theoretically secure and reveals no information other than the results. Furthermore, it requires only two broadcasts and scales linearly in the number of items for sale. Moreover, bidders who refuse to participate in this protocol to compute the aggregate demand can be disqualified, and the demand recomputed without them. If a bidder submits incorrect values during this protocol, then the

computed values r_j^t will be discovered to be incorrect. Although the process we describe cannot detect which bidder submitted incorrect values, the auctioneer can resort to a more sophisticated *verifiable* secret sharing protocol (e.g., [48]) that can identify non-compliant bidders. The ability to use such protocols if necessary should discourage malicious bidders from attempting to disrupt our protocol: they can always be discovered and disqualified. The auctioneer can also recover and reveal disqualified bidders' prior bids once he receives the time-lapse decryption key.

While this is a simple protocol for bidders within the auction to follow, other methods of computing aggregate demand are possible. One notable example is the threshold variant of Paillier cryptography advanced by Damgård and Jurik [51].

7.4.2 Transition to the Proxy Phase

Let T denote the number of rounds in the clock phase. Each bidder has submitted a bid on $\langle s_i^1, \ldots, s_i^T \rangle$ bundles at public prices $\langle p^1, \ldots, p^T \rangle$. A bidder can now:

(a) improve any bid submitted during the clock phase

(b) include bids on additional bundles

These additional bids are committed by each bidder, by encrypting with the key associated with the TLC service and then sharing them, for instance posting them to a public bulletin board. When the auctioneer receives the time-lapse decryption key he will then prove that each bidder meets the activity rules that constrain her ability to bid in this transition from clock to proxy.

For (a), we first require each bidder B_i to associate a bid price $b_i(s_i^t)$ with every

bid. This bid price must satisfy

$$b_i(s_i^t) \geq p^t \cdot s_i^t \tag{7.3}$$

For (b), each bidder can also submit additional bids, which we index $k > t$ to indicate that they are received after the close of the clock phase. Consider some bundle s_i^k, either one of the clock bundles or one of these additional bundles, and its associated bid price $b_i(s_i^k)$. Any such bid must satisfy the following constraints:

$$b_i(s_i^k) - p^t \cdot s_i^k \leq \alpha(b_i(s_i^t) - p^t \cdot s_i^t), \quad \forall t \in \{1, \ldots, T\} \tag{7.4}$$

This requires that the bidder would not have been much happier (by some relaxation parameter $\alpha \geq 1$) by bidding this bundle in any clock round than the bundle that it did bid in that round. We will also require each bidder to *pad* her bids (with zero bids), so that the total number of bundles that receive a bid is constant across all bidders. Let K denote the number of such bids.

Once this transition round closes the auctioneer receives the time-lapse decryption key and will now generate a proof that all bids satisfy these activity rules. If a bidder submits a non-compliant bid at this phase, the auctioneer can prove the bid is non-compliant using our framework and remove any such bids from the computation of the outcome.

To establish the activity rule, then for every bidder B_i and round $t \in \{1, \ldots, T\}$, the auctioneer computes provably correct encryptions of the dot products $p^t \cdot s_i^t$ for values bid during the clock phase. He further computes, for every bidder B_i, the $t(K-T)$ dot products $p^t \cdot s_i^k, \forall t \in \{1, \ldots, T\} \forall k \in \{T+1, \ldots, K\}$. These dot products are computed in the same way encrypted dot products are computed at the end of

Section 7.4.1. To prove Eq. 7.4, he shows that the bidder prefers each final proxy bid $\langle s_i^k, b_i(s_i^k)\rangle, T < k \leq K$, he computes the encrypted differences of these encrypted dot products and encrypted bid values $b_i(s_i^k)$ and $b_i(s_i^t)$ (respectively) and multiplies the second result by the public constant α; this allows him to use a simple interval proof to demonstrate the inequality.

7.5 Conducting the Proxy Auction

The proxy phase of the CCP auction is used to determine the final allocation of goods and the final payments. This requires solving a sequence of linear optimization problems. Given that the winner-determination problem for combinatorial auctions is NP-hard, it is essential that the bids are now revealed to the auctioneer. This will enable the auctioneer to leverage efficient methods of integer programming in determining the outcome. We reiterate that this revelation occurs after the auctioneer can influence any of the bids with this information.

We show how to use the homomorphic properties of cryptosystems to establish the correctness of an encrypted solution to an integer program. This is the main technical innovation. In particular, we work with *branch-and-bound* trees and explain how to use cryptographic methods to establish that a solution to an integer program is optimal by reference to establishing various linear constraints implied by a fathomed (i.e. solved) branch-and-bound tree. What we find appealing about our approach is that it is completely agnostic to the particular heuristics by which a branch-and-bound proof tree is generated (e.g. depth-first, breadth-first, memory management, branch-selection heuristics, etc.). Rather, the system works directly with the information

that is established upon conclusion of the search.

We confine our solution to what can be considered a standard, textbook treatment of branch-and-bound search (e.g., see Wolsey [155]). In doing so, we impose two main restrictions on the use of branch-and-bound algorithms: (a) no pre-processing, and (b) no cut-generation. While modern optimization solvers, such as ILOG's CPLEX, do make extensive use of both of these methods, good performance can be achieved on reasonably sized problems without either feature. Our mechanism is already orders of magnitude more efficient to verify than the methods described in earlier cryptographic combinatorial auction protocols. We reserve for future work further consideration of computational optimizations.

7.5.1 Branch-and-Bound Search

To illustrate the principle of branch-and-bound search we will consider the winner-determination problem (WDP) in the proxy phase. In defining this, we index the proxy bids $s_i = \langle s_{i1}, \ldots, s_{iK} \rangle$ from each bidder i. Recall that K is the total number of bids received from each bidder (by padding if necessary.) Let $b_i = \langle b_{i1}, \ldots, b_{iK} \rangle$ denote the associated bid values. The integer programming (IP) formulation for the WDP is

$$\max \left\{ \sum_i \sum_k x_{ik} b_{ik} \ : \ \text{s.t. } x \in \mathbb{F}, x_{ik} \in \{0,1\}, \ \forall i, \forall k \right\} \tag{7.5}$$

where

$$\mathbb{F} = \left\{ \begin{array}{cc} \sum_i \sum_k s_{ikj} x_{ik} & \leq C_j, \ \forall j \in G, \\ \sum_k x_{ik} & \leq 1, \ \forall i \in B \end{array} \right\}, \tag{7.6}$$

Bidder	Bid	Variable	Items	Price
1	1	x_{11}	$\{A, B\}$	3
2	1	x_{21}	$\{B, C\}$	3
3	1	x_{31}	$\{A, C, D\}$	3
4	1	x_{41}	$\{C, D, E\}$	2
5	1	x_{51}	$\{E, F\}$	4.5
6	1	x_{61}	$\{G\}$	3
7	1	x_{71}	$\{D\}$	1

Table 7.1: A simple example to illustrate a branch-and-bound tree

and these constraints ensure that no more units of a good are allocated than in the supply and that no more than one bid is accepted from any single bidder.

In describing branch-and-bound, let $\underline{z}$ denote the value of the best solution found so far (initialized to $-\infty$), and let $\underline{x}$ denote that solution (undefined when no solution has been found.) This is the *incumbent* solution. The first step in branch-and-bound is to solve the linear-programming (LP) relaxation,

$$\max \left\{ \sum_i \sum_k x_{ik} b_{ik} \ : \ \text{s.t. } x \in \mathbb{F}, x_{ik} \geq 0, \ \forall i, \forall k \right\} \tag{7.7}$$

Let $L^0 = \{x : x \in \mathbb{F}, \ x_{ik} \geq 0, \forall i, \forall k\}$ denote the LP-relaxation of the solution space. Let $\overline{x}^0$ denote the solution on L^0 and $\overline{z}^0$ the value of this solution. If $\overline{x}^0$ is *integral* then branch-and-bound can stop with $\underline{x} := \overline{x}^0$ and $\underline{z} := \overline{z}^0$. The solution $\overline{x}^0$ will in general be *fractional*, meaning that one or more of the variables has a value that is neither 0 or 1.

To make sense of this consider the simple example in Table 7.1 (adapted from Sandholm et al. [138]) in which we assume 7 bids, each from a unique bidder, and 6 goods all in unit supply. The optimal solution is to allocate to bids {1,5,7} for a total value of 8.5. But the solution to the LP relaxation is fractional, with an assignment

$\langle 0.5, 0.5, 0.5, 0, 1, 0, 0.5\rangle$ and total value of 9.5. When this occurs, a branching decision is made on one of the fractional variables. Continuing with the example, suppose that we branch on $x_{71} \leq 0$ and $x_{71} \geq 1$. This generates two new sub-problems, one defined on solution space $L^1 = \{x : x \in \mathbb{F}, x_{71} \leq 0,\ x_{ik} \geq 0,\ \forall i, \forall k\}$ and one defined on solution space $L^2 = \{x : x \in \mathbb{F}, x_{71} \geq 1,\ x_{ik} \geq 0,\ \forall i, \forall k\}$. Branch-and-bound continues by picking one of these and solving the associated linear program. Let $(L^p, \overline{x}^p, \overline{z}^p)$ denote the associated LP and solution. In any one of the following three cases, this becomes a *"fathomed"* (or solved) leaf:

(a) the subproblem is infeasible

(b) the subproblem has an integral optimal solution; if $\underline{z} < \overline{z}^p$ then $\underline{z} := \overline{z}^p$ and $\underline{x} := \overline{x}^p$.

(c) the subproblem is feasible and the solution fractional, but $\beta\overline{z}^p \leq \underline{z}$ for some $\beta \leq 1$ that controls the optimality tolerance.

In our example, the solution to L^2 is integral and we would set $\underline{z} := \overline{z}^2 = 8.5$ and $\underline{x} := \overline{x}^2 = \langle 1, 0, 0, 0, 1, 0, 1\rangle$. This leaf is now fathomed. But the solution to L^1 is fractional ($\overline{x}^1 = \langle 0.5, 0.5, 0.5, 0, 1, 0, 0\rangle$) and has value $\overline{z}^1 = 9 \not\leq \underline{z} = 8.5$. In such a case, branch-and-bound search will generate two additional subproblems, typically by doing something like branching on the most fractional variable. The unsolved subproblems are stored on the "open list." Branch-and-bound finally terminates when the open list is empty, returning the incumbent as the solution. Finishing with the example, when we branch on $x_{11} \leq 0$ and $x_{11} \geq 1$ we obtain two leaves that are fathomed. The LP relaxations generate integral solutions and their value is less than that of the solution already found.

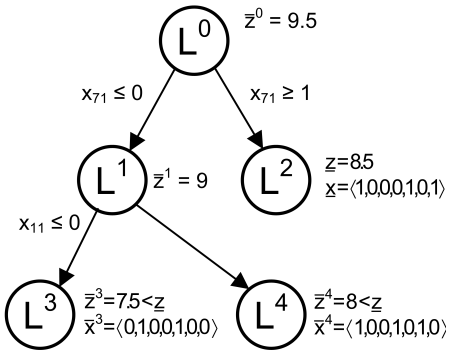

Figure 7.1: Branch-and-Bound Proof Tree

While there are many sophisticated strategies for managing the details of a branch-and-bound search, for our purposes all that is required is a *fathomed* branch-and-bound tree, i.e. one for which all leaves have been fathomed. An example of a so-called *proof tree* for the example is shown in Figure 7.5.1.

7.5.2 The General Approach

In this section we describe the general approach to establish the correctness of the solution to an integer program (IP). Along the way we also provide a method to establish the correctness of the solution to a linear program (LP). Recall that the *input* to the IP is published in encrypted form. In describing our approach we assume that the solution to the IP is revealed to all parties, but this is not necessary. All relevant steps can instead be performed using an encryption of the solution if the solution itself is to remain private.

The cryptographic proof is constructed around a proof tree as generated at the termination of a branch-and-bound search. To perform these steps on the encrypted inputs, we first note that IPs, LPs and their duals are all linear inequalities and

linear equations in the objective. Therefore, we can prove a set of constraints is satisfied, or that a solution has a particular objective value, using the verifiable addition, subtraction and multiplication operations and equality and inequality tests on Paillier-encrypted values. All that is required are encryptions of all the private inputs (the bids in our case).

Because we have formulated all inputs as integers, it is theoretically possible to obtain LPs with rational coefficients at every point in the proof tree, which implies that they have rational solutions. Thus our extension of integer arithmetic to the rationals (Section 2.1.5) enables us to calculate exact solutions to rational LPs. In practice, it is likely that the results will be computed using a computer program that yields a floating-point or real number as a result. We can instead convert this value to a rational number and prove that the constraints are satisfied with acceptably small error; see Section 2.1.7.

The proof of the correctness of a solution x^* to a IP proceeds with the following steps:

1. Any permutation-invariance in the class of problems being solved is leveraged for the purpose of secrecy by generating a random permutation using a mix network as described in Section 7.2.1. This proves to verifiers that the set of encrypted values in the proof tree is the same as the set of inputs, but makes the correspondence between those sets is unknown.[10]

2. The branching decisions that define the proof tree are revealed. (For instance,

[10] A complete permutation invariance is not required for this step. For example, in the context of the combinatorial auction application, we seek a permutation of the order of the bids submitted by a particular bidder and also a permutation across bidders. But we should not mix-up bids submitted by one bidder with bids submitted by another bidder.

"at the root the left branch is $x_6 \leq 0$ and the right branch is $x_6 \geq 1$" and so on.) The amount of information that this reveals depends on the amount of permutation invariance in the class of problems. For example, if all inputs can be "mixed" with all other inputs then this reveals no information.

3. The solution x^* to the IP is revealed along with a claim $\beta \leq 1$ about its optimality (e.g., $\beta = 9999/10000$ would state that the solution quality is within multiplicative factor 9999/10000 of the optimal solution.) The encrypted solution $E(x^*)$ is published and shown to be a valid encryption of x^*: this is because many of our operations only apply to two encrypted operands, and for those we need to use $E(x^*)$ rather than the unencrypted x^*.

4. Let q^* denote the leaf associated with the optimal solution. This is revealed by the prover. The prover then proceeds to:

   (a) Publish $E(V^*)$ and prove that its value is correct (i.e. the value is an encryption of the objective value of the IP given solution x^*).

   (b) Prove that x^* satisfies the constraints of the LP formulated at leaf L^{q^*} (i.e. prove inequalities defined in terms of the encrypted input to the IP and also the additional inequalities implied by the branching decisions.)

   (c) Prove that x^* is integral. (See Section 2.1.5.)

5. Consider every leaf q (including the optimal leaf) in turn. For every such leaf, the prover then proceeds to:

   (a) Let y^q denote the solution to the dual LP at leaf L^q and D^q the value of that

dual solution. Publish the encrypted dual $E(y^q)$ solution and the encrypted dual value $E(D^q)$ at this leaf.

(b) Prove that the dual solution satisfies the constraints of the dual LP formulated at leaf L^q.

(c) Prove the correctness of the dual value $E(D^q)$ by reference to the dual formulation, and that $\beta E(D^q) \leq E(V^*)$.

This procedure encompasses both leaves that are fathomed by infeasibility and leaves that are fathomed by bound in the same way. Note that a leaf that is infeasible in its primal form has a dual solution with value $-\infty$ by the duality theory of LP. Therefore, the prover can always construct a feasible dual solution to prove that there is no better (primal) solution in the feasible solution space that corresponds to a particular leaf. It should be easy to see how to generalize the above approach to a mixed integer program.

When the original problem is an LP rather than a IP then there is no proof tree to deal with, and the procedure is simply: (a) publish $E(V^*)$ and prove this value is correct; (b) prove that x^* satisfies the constraints of the LP; (c) publish an encrypted dual solution $E(y^q)$ and associated dual value $E(D^q)$; (d) prove that the solution is dual feasible, and that $\beta E(D^q) \leq E(V^*)$.

7.5.3 Winner Determination

To instantiate the general approach in the context of the proxy auction we provide the IP formulation for the winner-determination problem (WDP) along with the dual problem to the LP relaxation at a leaf of a branch-and-bound tree. Recall that

(s_{ik}, b_{ik}) denotes the kth proxy bid submitted by bidder i, where bundle s_{ik} contains s_{ikj} units of item $j \in G$. The IP formulation for the WDP is:

$$\max_{x_{ik}} \sum_{i \in B} \sum_{k} x_{ik} b_{ik} \qquad \text{WDP(B)}$$

$$\text{s.t.} \quad \sum_{i \in B} \sum_{k} s_{ikj}\ x_{ik} \leq C_j, \quad \forall j \in G \qquad (7.8)$$

$$\sum_{k} x_{ik} \leq 1, \quad \forall i \in B \qquad (7.9)$$

$$x_{ik} \in \{0, 1\}, \quad \forall i \in B, \forall k$$

where x_{ik} indicates whether the kth bid from bidder i is accepted. We label this formulation WDP(B) to make explicit that this is problem is defined for all bidders and to allow for variations WDP(L) defined on a subset $L \subseteq B$ of bidders. Constraints (7.8) ensure that the supply constraints are satisfied. Constraints (7.9) ensure that no bidder receives more than one bundle of items.

The linear-programming relaxation of WDP(B) is defined by replacing $x_{ik} \in \{0, 1\}$ with $x_{ik} \geq 0$. In defining the dual (and overloading notation from the clock phase, which is no longer needed), we introduce variables p_j to denote the dual variable for constraints (7.8) and π_i to denote the dual variable for constraints (7.9). Given this, then the dual problem is:

$$\min_{p,\pi} \sum_{j} C_j p_j + \sum_{i} \pi_i \qquad \text{DWDP(B)}$$

$$\text{s.t.} \quad \sum_{j} s_{ikj}\ p_j + \pi_i \geq b_{ik}, \quad \forall i, k \qquad (7.10)$$

$$p_j \geq 0, \pi_i \geq 0$$

A sequence of branching decisions leading to a fathomed leaf in the search tree

introduces additional constraints to WDP(B) and modifies the dual problem at the leaf. Let $(i,k) \in OUT$ indicate that branch $x_{ik} \leq 0$ has been taken and $(i,k) \in IN$ denote that branch $x_{ik} \geq 1$ has been taken. Given these constraints, the restricted primal and dual pair becomes:

$$\max_{x_{ik}} \sum_i \sum_k x_{ik} b_{ik} \qquad \text{RWDP(B)}$$

$$\text{s.t.} \quad \sum_i \sum_k s_{ikj}\ x_{ik} \leq C_j, \quad \forall j \in G \qquad (7.11)$$

$$\sum_k x_{ik} \leq 1, \quad \forall i \qquad (7.12)$$

$$x_{ik} \leq 0, \quad \forall (i,k) \in OUT \qquad (7.13)$$

$$x_{ik} \geq 1, \quad \forall (i,k) \in IN \qquad (7.14)$$

$$x_{ik} \geq 0, \quad \forall i, \forall k$$

$$\min_{p,\pi,\delta} \sum_j C_j p_j + \sum_i \pi_i - \sum_{i|(i,k)\in W} \delta_i \qquad \text{DRWDP(B)}$$

$$\text{s.t.} \quad \sum_j s_{ikj}\ p_j + \pi_i \geq b_{ik}, \quad \forall (i,k) \notin (OUT \cup IN) \qquad (7.15)$$

$$\sum_j s_{ikj}\ p_j + \pi_i - \delta_i \geq b_{ik}, \quad \forall (i,k) \in IN \qquad (7.16)$$

$$p_j \geq 0, \pi_i \geq 0, \delta_i \geq 0$$

Dual variable δ_i corresponds to constraints (7.14) in RWDP(B). The variable that dualizes constraints (7.13) drops out of the dual formulation because it appears with coefficient zero in the objective and appears in a non-binding constraint. Taken together with the general approach of Section 7.5.2, we now have all the pieces to

see how to establish for a verifier the correctness of the allocation determined in the proxy phase. Once the solution x^* is published and associated with a leaf of the branch-and-bound tree, and once it has been shown to satisfy the constraints of the appropriate RWDP(B) formulation for that leaf and to be integral, the remaining work in proving the optimality is in terms of the DRWDP(B) formulations at each leaf. All the information required to complete these proofs is either available in the encrypted proxy bids (e.g. s_{ikj}, b_{ik}), publicly known (e.g. the capacity C_j), or defined by the branching decisions (i.e., $\{OUT, IN\}$).

7.5.4 Proxy Payments

The payments in the proxy auction are those on the buyer-optimal core that minimize the maximal deviation across all buyers from the VCG payoff profile.[11] Solving for this point will require using constraint generation but the cryptographic proof will be constructed after-the-fact in terms of just the final set of constraints. By a slight reformulation of the method in Day and Raghavan [52], the payoffs to

[11]This is a particular instance of a family of payment rules that break ties in different ways but have very similar economic properties.

winning buyers $i \in W$ can be computed in the following LP:

$$\max_{\pi,m} \sum_{i \in W} \pi_i - \epsilon\, m \qquad \text{EBOP}$$

$$\text{s.t.} \quad \sum_{i \in W \setminus L} \pi_i \leq V^* - V(L), \quad \forall L \subseteq W \tag{7.17}$$

$$\pi_i + m \geq \pi_i^{\text{vcg}}, \quad \forall i \in W \tag{7.18}$$

$$0 \leq \pi_i, \quad \forall i \in W$$

$$0 \leq m,$$

with $\pi_i = 0$ for all $i \notin W$, and for some small $\epsilon > 0$. The objective is to maximize the total buyer payoff, but then for small ϵ to break ties in favor of minimizing the maximal deviation m from the VCG payoffs across all such buyers. Constraints (7.17) are the core constraints and constraints (7.18) force m to adopt the maximal difference to VCG payoffs. Given a solution π^* to EBOP, the payments collected from each winning buyer $i \in W$ are $b_i(s_i^*) - \pi_i^*$.

EBOP is an LP and has no integer variables. But notice that part of its input has required solving IPs (since constraints (7.17) are defined in terms of V^* and $V(L)$). More difficult, there are an exponential number of constraints (7.17). Day and Raghavan [52] suggest using *constraint generation* to construct a subset $\mathcal{L} \subseteq 2^W$ of coalitions, with constraints (7.17) reformulated as $\sum_{i \in W \setminus L} \pi_i \leq V^* - V(L), \quad \forall L \in \mathcal{L}$. Let EBOP($\mathcal{L}$) denote the relaxed form of EBOP in with just this subset of constraints. New constraints are introduced until it can be established that:

$$\max_{L \subseteq W} \sum_{i \in W \setminus L} \pi_i - (V^* - V(L)) \leq 0 \tag{7.19}$$

This establishes that none of the missing constraints is binding. (In practice, this

is also the separation problem that is solved in generating a new constraint.) Given a solution π^* to EBOP($\mathcal{L}$), the separation problem can be formulated and solved via an IP as a simple variation on the regular WDP:

$$\max_{x_{ik}} \sum_{i \in W} \left(1 - \sum_k x_{ik}\right) \pi_i - V^* + \sum_{i \in W} \sum_k x_{ik} b_{ik} \qquad \text{SEP}(\pi^*)$$

$$\text{s.t.} \quad \sum_i \sum_k s_{ikj}\, x_{ik} \leq C_j, \quad \forall j \qquad (7.20)$$

$$\sum_k x_{ik} \leq 1, \quad \forall i \in W \qquad (7.21)$$

$$x_{ik} \in \{0, 1\}$$

Putting this all together, the methodology for establishing the correctness of the final payments is as follows:

1. Publish the set $\mathcal{L}$ of coalitions of winners that are used to establish the correctness of payments. (Note that this does not reveal any information if a mix network was used on the inputs.) Publish the parameter $\epsilon > 0$.

2. Publish the solution $E(\pi^*)$ and $E(m^*)$ to EBOP($\mathcal{L}$). Publish the vector of proxy payments $p^* = \langle p_1^*, \ldots, p_n^* \rangle$. Prove that $p_i^* = \sum_k x_{ik}^* b_{ik} - \pi_i^*$ for all buyers i.

3. Publish and establish the correctness of $E(\pi^{\text{vcg}})$, for $\pi^{\text{vcg}} = \langle \pi_1^{\text{vcg}}, \ldots, p_n^{\text{vcg}} \rangle$. Publish and establish the correctness of $E(V(L))$ for all $L \in \mathcal{L}$.

4. Publish and prove the solution to the separation problem SEP(π^*).

5. Prove that the solution to EBOP is primal feasible.

6. Publish an encrypted solution to the dual problem and prove it is dual feasible. Prove the value $E(D^*) \leq \beta E(V^*)$ for some parameter $\beta \geq 1$, e.g.

$\beta = 100001/100000$.

Step 3 requires proving facts about solutions to different winner determination problems. For the VCG payoff, $\pi_i^{\text{vcg}} = V^* - V(B \setminus i)$ and thus this needs the value of $E(V(B \setminus i))$ to be proved correct. This can be done following the approach in the previous section for the WDP. Similarly, we need to prove the correctness of $E(V(L))$ for subsets $L \subseteq B$. Note that both kinds of proofs can be verified without revealing the solution to these subproblems, and that no useful information leaks from publishing branching decisions in the branch-and-bound search because of the use of a mix network.

Step 4 can be reduced to an instance of the WDP and proved analogously. To see this, notice that the objective can be reformulated as

$$max_{x_{ik}}(\sum_{i \in W} \pi_i - V^*) + \sum_{i \in W} \sum_{k} x_{ik}(b_{ik} - \pi_i), \tag{7.22}$$

and that $(\sum_{i \in W} \pi_i - V^*)$ is a constant. Thus, this is a winner determination problem formulated on the winners W and with the bid values of the winners replaced with adjusted values $b_{ik} - \pi_i$. In Step 6 we need the dual to EBOP($\mathcal{L}$). Introducing variables z_L and z_i for the generated subset of constraints (7.17) and constraints (7.18) respectively, the dual LP is:

$$\min_{z_L, z_k} \sum_{L \in \mathcal{L}} (V^* - V(L)) z_L + \sum_{i \in W} \pi_i^{\text{vcg}} z_i \qquad \text{DEBOP}(\mathcal{L})$$

$$\text{s.t.} \quad \sum_{L: i \notin L} z_L \; - z_i \geq 1, \quad \forall i \in W \tag{7.23}$$

$$- \sum_{i \in W} z_i \geq -\epsilon \tag{7.24}$$

$$z_L, z_i \geq 0$$

The correctness of the solution to EBOP($\mathcal{L}$), which is a linear program rather than an IP, can then be verified by following the specialized methodology outlined at the end of Section 7.5.2.

7.5.5 Announcing Results

The above steps are sufficient to prove to any interested party that the allocation and payments are correct. But because we employed a mix network to prevent bidders from learning the position of their bids in the proof tree, we still need to convince an individual bidder that the particular allocation announced for them is correct for them.

This is easy to achieve by privately revealing to each bidder *only* the correspondence between their original proxy bid that was accepted and its position in the permutation generated by the mix network. The bidder will then be satisfied that the outcome proven is correct from her perspective because she can verify that her bid was allocated in the optimal allocation. She will similarly believe that the payment corresponding to the bidder that submitted the bid, and hence her payment, is correct. We note that this does now imply an extremely tiny amount of information that is leaked by our system, over and above that implied by the outcome of the auction. Namely, each bidder learns where in the various proof trees her own accepted bid was branched on. But this appears to disclose no useful information to the bidder.

7.6 Conclusions

We have described a scalable cryptographic method to enable trusted but secret combinatorial auctions. In particular, we have fully captured the intricacies of the clock-proxy auction in our solution. It bears additional emphasis that in striving for a practical solution we require that the auctioneer is trusted not to reveal information about bids after the auction closes. This is the same tradeoff that we made in our earlier work on non-combinatorial auctions [119]. In making this tradeoff, we achieve a system that is provably correct and trustworthy, and we believe computationally and commercially practical.

Chapter 8

Cryptographic Securities Exchanges

While transparency in financial markets should enhance liquidity, its exploitation by unethical and parasitic traders discourages others from fully embracing disclosure of their own information. Traders exploit both the private information in upstairs markets used to trade large orders outside traditional exchanges and the public information present in exchanges' quoted limit order books. Using homomorphic cryptographic protocols, market designers can create "partially transparent" markets in which every matched trade is provably correct and only beneficial information is revealed. In a cryptographic securities exchange, market operators can hide information to prevent its exploitation, and still prove facts about the hidden information such as bid/ask spread or market depth.

8.1 Introduction and Related Work

Market information plays a crucial role in modern securities exchanges. Published trades inform the public about the value of a particular security. Bid and ask quotations in limit books inform traders about other traders' interest in a security and at what prices orders are likely to be filled. Price change and trading volume information for equities track the (mis)fortunes and public awareness of corporations and equities markets. In theory, this market information should all benefit traders by forcing traders who have private information to disclose it via their trades. Unfortunately this information can also facilitate parasitic and unethical trading practices, and simple nondisclosure can itself lead to new exploits by market insiders who can benefit from the now private information. Balancing these forces is a significant challenge in market design, and our cryptographic tools offer an attractive solution to this problem: market designers can achieve unprecedented control over deciding exactly what must be transparent, and the ability to prove that what is revealed is correct.

The application of similar cryptographic tools in other commercial protocols has been well studied in the academic literature (open and sealed-bid auctions [94, 157], electronic cash [64], etc.) Yet, surprisingly little has been written about the contributions cryptography can make to securities markets, in particular the *open call auction* and *continuous double auction* protocols that underly most modern securities exchanges. Important prior work in this area includes Giovanni Di Crescenzo's pioneering work exploring privacy for stock markets [53], the secure double auction protocols Wang et al. employs homomorphic ElGamal encryption in [154], and Matsuo and Morita describe a "Secure Protocol to Construct Electronic Trading" in [98].

The work of Bogetoft et al. in [26], based on secure multiparty integer computation, proposes an application to securities exchanges, although in their protocol, "all trade is executed at the same market clearing price" and orders that do not clear are rejected. In our case, we wish to support both market orders and the limit order book that is an integral component of modern financial markets. Szydlo's work on zero-knowledge proofs for disclosure of portfolio risk [148], more relevant to Chapter 9 than this one, proposed the use of homomorphic cryptographic commitments to the disclosure of stock portfolio holdings. Although some of this related work, particularly [53], considers the privacy of trader identities, our own work concerns only the revelation of quantitative information about trades—not the anonymity of the traders—which we view as an orthogonal problem.

While the objective in most cryptographic work for auctions has been to hide information (secrecy), our objective is to enable a market designer to combine an appropriate level of *partial* transparency with provably correct behavior. We also do this in a setting informed by real-world demands, specifically, an exchange with both limit and market orders, and in which multi-party computation by all parties is infeasible.

Our design allows market designers to specify exactly what they wish to reveal, and reveal only that information while proving it, and the market operation, are correct. Immediate applications of our work can be seen in preventing unethical and parasitic trading practices in the major exchanges as well as providing for a means for trading large block orders without revealing information that can be exploited. Evidence for the need for information hiding in markets can be seen by recent SEC

investigations and criminal convictions of unethical traders, and the development of new alternative trading systems (ATS's) that privately match large block trades. We detail how market information is misused and the securities industry's responses in Section 8.2.1.

In situating our work, we first discuss the role of information in securities markets from the perspective of *market microstructure*, a rich area of financial research that studies the role and exchange of information in markets and how market design principles serve to foster or inhibit information exchange. Market microstructure studies questions such as: What are the costs and benefits of transparency in financial markets? What determines the bid-ask spread for a particular stock? Do large orders really move the market? What is the effect of (not) publishing insider trades?

For simplicity, we will consider a single, *electronic clearing network* in which specialists, broker/dealers, or retail traders may place limit or market orders for shares of a particular equity (e.g. IBM stock) in a continuous double auction. We will explain the roles of each of these parties in the market, the types of transactions they may participate in, and why they do. We consider the various forms of information that these parties reveal through their actions (or inactions) and what information the markets reveal to them, and how they can profit from that information.

After considering the role of participants and information in our market, we construct a cryptographic framework that enables this information to be finely controlled and disseminated according to the specific rules established by a market operator. We observe that presently these types of information control are not achievable in financial markets because of a lack of trust: it is this transparency that proves *correctness*

of the market transactions. Yet requiring full *transparency* to achieve correctness is a blunt method that can be exploited. We decouple these considerations.

Our proposed system proves correctness and provides for any level of transparency; being able to prove facts without directly revealing the numbers behind them offers market designers a more expressive set of possibilities for reporting market status. Our construction and protocols use the homomorphic cryptography describe in Chapter 3 and several related works [114, 51, 115, 154, 148] to prove the correct operation of the market according to its published rules and also to credibly reveal the required market information to the participants. Another advantage of our solution is that unencrypted quotes from the primary market can easily be integrated into our encrypted order book and matched against standing block limit orders; there is no need to fragment orders into different market centers.

It is not our intention to advocate particular kinds of transparency but rather to offer a finer level of control to market designers. Indeed, this application of cryptography seems to us to open up interesting new questions for the field of finance. We conclude with worked examples and report the result of an initial analysis of the cost to support a realistic order flow on current hardware.

8.2 Introduction to Securities Exchanges

In this section we provide an overview of how equities are traded in order to motivate our contributions to those without a background in finance. The study of market microstructure in finance is most applicable to our work; Larry Harris' book *Trading & Exchanges* [73] is a well respected textbook on the field; we also found

three recent survey papers of market microstructure [24, 95, 144] helpful in framing our contributions.

In many cases, we will simplify the complex workings of modern financial markets in order to illustrate the core principles that are relevant to our work. We clearly indicate these simplifying assumptions in our exposition. We use as our model a market for a single equity for a single company and assume that all trades in that market take place on an electronic clearing network (ECN) running a continuous double auction with an open limit order book. We assume that the market operates at fixed daily opening and closing times and trading does not take place anywhere else when the market is closed. For simplicity, we do not consider short sales or buying to cover, which are equivalent to selling and buying long positions for our purposes.

The market maintains an *order book* in which all outstanding limit orders are recorded. Depending on the transparency rules of the market, all, some, or none of the limit orders on the order book may be available to the public. Real-world exchanges (NYSE, NASDAQ, Chicago Board of Trade) offer various degrees of transparency for their order books.

For the purposes of the cryptographic properties of our exchange, there is no important difference between dealers, brokers, specialists, or investors. In our simplified model, everyone may post limit orders and has access to the same information.[1] Therefore we only consider two classes of participants: the (market) *operator*, i.e. the exchange or its agent, and *traders*, in which we include specialists, broker/dealers, and institutional and retail investors.

[1] A simplified way to look at it is that dealers, brokers and specialists provide liquidity to the market to support the trades investors want to make. Possibly the most important function of liquidity providers' use of limit orders is enabling investors to place market orders.

As we present our model, we introduce formal definitions that we use later in our protocol construction. We model the market state as the current state of the limit order book B and trade history H. The order book B is private, but the trade history H is public. (H can be public because any values logged therein are maintained in encrypted form.) The market operator also maintains a public, encrypted order book $\hat{B}$ that is equivalent to B except that all bids and quantities are encrypted. Each order placed receives a unique identifier i regardless of whether it is a bid or ask order which is associated with the order and its components. Ask and bid orders, a_i and b_i respectively, enter the market when placed and exit the market when withdrawn or executed. An order is the tuple $(p_i, q_i, t_i, s_i \in \{\mathtt{a}, \mathtt{b}\})$ representing the price, number of shares, the time the order was placed on the market, and the side of the market: whether the order is an ask (sell) or a bid (buy). When an order is taken off the order book, it is removed from the state of the order books B and $\hat{B}$ and the history H is updated with the execution or cancellation that resulted in its removal.

We will also refer to a function with access to a complete price ordering of the orders on the market $o(s \in \{\mathtt{a}, \mathtt{b}\}, \mathit{rank})$ whose arguments are the side of the market (ask or bid) and the order's *rank* where the most competitive price on either side has $\mathit{rank} = 0$. Its output is the unique identifier i for the ask or bid with the given *rank*. For example, we might write the current bid/ask spread as $p_{o(\mathtt{a},0)} - p_{o(\mathtt{b},0)}$, or the market depth (measured in shares) of the most competitive ten bid orders as $\sum_{r=0}^{9} q_{o(\mathtt{b},\mathit{rank})}$. This ordering is maintained by the market operator; it is convenient for showing how the market operator proves correct operation of the market. Obviously, the ordering $o(s, r)$ changes whenever an order enters or exits the market. This function is also

used to support the market invariant that all orders are maintained in strict priority order as described in Section 8.3.

In modern equities markets, orders fall into two basic categories: *market* orders are an instruction to buy or sell a specific quantity of a security, and are filled as soon as possible at the best available price on the market; *limit* orders are an instruction to buy or sell a specific quantity of a security at a specific price, and are filled only when another participant in the market is willing to make the opposite trade.

More complex orders that use real-time market information are possible, depending on broker support; for example, a *stop loss* order at a particular price instructs the broker to sell a position at the market when the market reports a trade at or below that price.[2] In practice, some traders also use orders based on real-time data as a substitute for limit orders because of the information revealed by limit orders or to get a better price; for example, an order such as "Buy 1,000 shares at the market if there are any trades below $20.00" might be used instead of a limit order to "buy 1,000 shares at $19.99" in order to keep the trader's intentions secret and potentially get a better price if the stock price dropped sharply. It might be that in a partially transparent market in which limit order prices are hidden, traders would be more inclined to use limit orders in these cases.

[2]Limit orders cannot substitute for stop orders. Limit orders are persistent, and less competitive than the current equilibrium price; stop orders react to market movements and are at *more* competitive prices. Our framework can be extended to support stop orders with concealed prices; the operator would maintain a side list of stop orders and prove when their target prices are reached by executed trades, all without revealing either the trade price or stop order price.

8.2.1 Market Information and Its Misuse

In this section we explore how transparency can be exploited, then examine at a high level the types of market information whose transparency may be regulated by cryptographic systems.

Misuse of Market Information.

The information provided by transparency can be exploited by unethical or creative traders. To illustrate this hidden cost of transparency, we detail two common practices, one unethical and the other "parasitic": front-running and penny-jumping, respectively. Larry Harris' chapter "Order Anticipators" in *Trading & Exchanges* explores these and other related practices in depth [73]. We speculate that these exploitations of transparency may be part of the cause for the conflict between published theoretical market microstructure results that show transparency should improve liquidity and other empirical results that are ambiguous with respect to this question [130].

"Front-running" is the unethical practice of a party with private information about an incoming large order to the market running in front of that order to take a position in the hope of making a quick profit when the large order arrives. For example, a trader knowing that a mutual fund is going to buy a $10M position in IBM stock might buy a smaller position beforehand with the expectation that the mutual fund's purchase will drive the price higher. Front-running and allegations of it are widespread. In 2001, Dreyfus agreed to pay $20.5 million to settle accusations that their fund manager Michael Schonberg engaged in front-running [1]. In 2003, the

NYSE announced its Enforcement Division had investigated several specialist firms for rule violations including front-running and decided to bring disciplinary action against them. Seven specialist firms agreed to pay over $200 million to settle charges brought as a result of these allegations [2]. In July 2006 a Manhattan jury convicted former specialist firm Van der Moolen managers Michael Stern and Michael Hayward of fraud for trading stocks on the firm's account before filling clients' orders in order to boost Van der Moolen's profits and their own compensation [35]. In early 2007, *The New York Times* reported other allegations of front-running: "The [SEC] has begun a broad examination into whether Wall Street bank employees are leaking information about big trades to favored clients..." [8]. In a final example that lends credibility to our claim that cryptography is important to the operation of the markets we propose, Citi, Merrill Lynch and Lehman ex-traders were prosecuted for eavesdropping on traders' communications and profiting on that private information [141].

"Penny-jumping" is not illegal, but Harris describes the practice as "parasitic" [73] because it exploits market information, but traders who engage in the practice do not actually contribute new information to the markets. Specifically, a trader identifies a large limit order on the order book (e.g. 25,000 shares at $25.00) and places a smaller limit order one tick above that order (e.g. 1,000 shares at $25.01). The penny-jumper's order will be filled first, and he expects that, in the short term, his upside is greater than his downside, because his downside is protected by a free trading option created by the large limit order while his upside is theoretically unlimited. If following price movements are locally random, it is likely that the stock will trade at a higher price before the large order is filled. Moreover, that reasoning ignores the possibility that

a large "buy" order might signal other traders that the stock is desirable and thereby have an upward effect on its price. If the price happens to decline before it increases, the large order will be filled, and the penny jumper exits his position via the large order for a one-tick loss (e.g. after 20,000 of 25,000 shares have been filled).[3]

There is also evidence that knowledge of large ("block") trades is similarly exploited. According to Stoll [144], large blocks of stock are not sent to the open market because "The risk of pre-trading portions of the block in this manner is that other traders will become aware of the block and will sell in anticipation, perhaps driving the price down..." and because other traders can exploit knowledge of large orders in other ways (as above). While Stoll further claims that "empirical evidence of block trades is quite mild," Keim and Madhavan [80] (as cited in [95]) find in an empirical study that the average (one-way) price impact for a seller-initiated transaction is -10.2% from a benchmark three weeks before a large block trade, after adjustment for market movement.

Historically, block trades are filled by "upstairs markets" where brokers shop around by calling other brokers for the best deal. Keim and Madhavan "attribute this large price impact to information 'leakage' arising from the process by which large blocks are 'shopped' in the upstairs market." [95] The reason for hiding information in block trades is mainly to protect the traders before the large transaction occurs.[4]

[3]There is another downside risk: the limit order may be canceled at any time, eliminating the free trading put option.

[4]Gemmill [66] offers an empirical analysis consistent with this view of the effects of post-trade reporting of block trades on the London Stock Exchange. He finds *ex post* disclosure of block trades does not have a dramatic effect on liquidity.

Responses to the Block Trading Problem

A common response to concerns of unethical and parasitic practices is to construct a market in which partial or no information on orders is reported; many of these exchanges, called "dark pools", have found some success but for others, institutional, liquidity-focused investors remain justifiably wary of some of them [78].

In the past few years, a number of major banks have collaborated with or built new ATS's (Alternative Trading Systems), generally with the purpose of making block trading more efficient. In late 2006, Citi, Goldman Sachs, Lehman Bros., Merrill Lynch, Morgan Stanley and UBS launched a "Block Interest Discovery Service" (BIDS) for automatically matching large block orders without revealing them to the primary markets. Around the same time, Citi, Lehman Bros., and Merrill Lynch also joined Credit Suisse and Fidelity in launching LeveL, another ATS. Still another clearing network, Liquidnet, specializes in helping institutions find counterparties for pairwise large block trades and has captured a small but significant share of order flow. Goldman Sachs' SIGMA platform consolidates liquidity from many parties and claims this gives clients better execution. NYSE Euronext plans a 2008 launch of a similar electronic block trading platform for European markets [37].

Each of these systems is designed with three purposes: first, they seek to provide block traders with the opportunity to trade with minimal market impact; second, they keep traders' identities anonymous; and third, they keep information about intended trades as secret as possible until the trade is executed. The BIDS platform also seeks to mitigate the dissemination of information by revealing trading intentions only after two parties have presented legitimate opportunities to trade. Another approach

is taken by Pipeline, which maintains an electronic "board" of the most commonly traded equities. Equities light up when there is block trading interest, but do not reveal whether the interest is to buy or sell. Blocks are traded in extremely large units (e.g. 10,000 or 25,000 shares).

Liquidnet's matching system is integrated into traders' order management systems and identifies opportunities for two parties to trade based on existing liquidity. Liquidnet also provides an important role as a trust broker: only parties who are known to be trading in good faith for liquidity reasons are permitted to participate in the Liquidnet network. Liquidnet bans traders who are believed to exploit the system, so that traders who use Liquidnet are less fearful of Liquidnet's knowledge of their orders [77].

POSIT, a service of Investment Technology Group, features block trading services that more closely resemble an exchange, in that it runs both scheduled matches and ongoing crossing of block trades. (Its BLOCKalert system is similar to Liquidnet, by integrating with order management systems to identify trading opportunities.) For its scheduled matches, POSIT Match^{SM} takes orders throughout the day and clears the market at specific times. Orders that cross are filled at the midpoint of the bid-offer spread on the primary market, so the system does not need to consider prices. While its protocols are similar to those we propose, POSIT makes no guarantees of execution and provides no correctness proofs—the systems rely completely on trust.

Several problems remain with these unregulated dark pools. First, if prices are hidden, an unscrupulous market operator could simply fill a favored party's bid before higher bids. Indeed, regulators have begun to mandate transparency to protect

investors, in light of specialists and broker/dealers exploiting private market information to their advantage [144, 95, 2, 1, 35]. Second, traders with private information may seek to extract liquidity from unknowing institutitonal investors participating in the dark pools. Or, traders may seek to discover and front-run block orders in a dark pool by placing probe orders designed to search for liquidity. Since institutions place limit orders with no guarantee of whether they will be filled, those who do place large orders may find that that information is discovered by strategies that search for liquidity by placing many small orders that sense a large block.[5] Some ATS's, for example, Pipeline, enforce a strict, large block order size to prevent such attacks.

The popularity of and growth in ATS's is clear: LiquidNet reported on its website in May 2008 that its network traded an average of 80 million shares per day in the first quarter of 2008. Pipeline traders exchanged 36 million shares in January 2008, also according to its website. These numbers have been increasing by double digit percentages year over year, reflecting the increased interest in block trading and the need for services that hide exploitable information.

While alternative trading systems certainly reduce the exploitation of order information, they do not provide any correctness guarantees. These approaches also typically require that the block trades be separated from the primary securities exchanges. This has had a fragmenting impact on securities trading: because there are so many ATS's where liquidity might be lurking, sophisticated computer engines now search across many sources of (often "dark") liquidity in order to get the most competitive prices [78]. Many bulge bracket firms boast an aggressively-named offering in algorithmic trading that seeks out liquidity from dark pools, including Credit

[5]See [78] for a detailed discussion of such "probe orders" and similar attacks.

Suisse's "Guerilla" and "Sniper", Citi's "Dagger", and Banc of America Securities' "Ambush" and "Razor".

Although the present work ignores privacy in these transactions, commercial solutions typically highly value trader anonymity. This indicates that cryptographic research in maintaining trader privacy (e.g. Di Crescenzo [53]) may also find a warm reception in the industry.

8.2.2 Developing a Cryptographic Securities Exchange

Our model is of a simple securities exchange in which a market operator keeps a private order book B and publishes its public analog $\hat{B}$ with encrypted prices and quantities, and (optionally encrypted) history of its actions H. Incoming limit orders are placed on the book or matched with existing limit orders; incoming market orders are matched with limit orders; the operator proves its actions correct.

Our primary goal is to prevent various adversaries from exploiting information present in limit order books to the detriment of traders who wish to place limit orders. These adversaries primarily include other traders and market insiders (market makers, specialists, exchange employees) who attempt to (unethically or parasitically) profit by exploiting limit order information.

Rindi [130] uses the term "partial transparency" in her examination of three regimes of pre-trade transparency in a market for a risky asset based on an open limit-order book: "under full transparency agents can observe the order flow and traders' personal identifiers; under partial transparency they can observe the order sizes and under anonymity they can only observe the market price."

We consider each of these information classes in turn. For existing orders, the type of the order is implied; if it is in the book, it is a limit order. Incoming orders may be market or limit orders; we assume that is disclosed. In call auctions, the transaction type (buy or sell) can be kept secret until the auction closes, but it is not meaningful to hide whether an order is to buy or sell in continuous double auctions. As noted before, timed expiration of orders is unimportant.

The price per share p_i associated with an order a_i or b_i on the book may be fully transparent ($p_i = \$20.06$), partially transparent ($\$20.00 \leq p_i \leq \$20.25$), or kept completely private ($p_i =?$). Similarly, the quantity q_i and time posted t_i may be fully, partially, or not transparent.

The parameters of multiple orders may be related by inequalities. Two orders may be related by price (e.g. $p_i \geq p_j$), quantity (e.g. $q_i = q_j$) or time posted (e.g. $t_i < t_j$). Partial or complete orderings for price and time of all orders in a limit book can be constructed using these methods, as will become important for more expressive partially transparent revelations. Quantity becomes important when proving order flow and correct execution of trades.

Finally, one might wish to prove information about linear functions on the parameters of multiple orders, or compute linear functions without revealing unnecessary additional information about the orders themselves. Examples of these functions include:

- Bid/ask spread between the two most competitive orders
- Market depth within p cents of the mean between the outstanding bid and ask (measured in number of shares)

- Bid-ask spread between the two least competitive orders comprising a market depth of q shares
- Prices (if any) at a market depth of q shares
- Average number of hours outstanding orders above price p have been on the market

Using recent advances in homomorphic encryption, market designers can construct markets in which this kind of information can be revealed and proved correct without revealing additional information about underlying orders.

Information, and related proofs, need not be issued in real-time, and in fact in many cases market designers may prefer delayed revelation. In our system, market designers can decide exactly when to reveal market activity, and even construct different disclosure rules for different trade sizes. For example, the market might disclose small trades within 30 seconds and large trades within 1 day.

8.3 The Cryptographic Securities Exchange

We have described the model of our market as a limit order book with a history. We consider the state of the order book B, the encrypted public order book $\hat{B}$, and the history H to be the core state of our market. Various actions by the participants in the markets update this state. We formally define these actions, who may perform them, and how the update the state of the market depending on its state. The order book and history begin as empty states.

In our present model we maintain an important invariant in B and $\hat{B}$: all orders are maintained in a strict priority ordering as defined by the ordering function $o(s, rank)$. Despite regulations that prescribe order routing priority, the priority of trades within active markets is a complicated process beyond the scope of the present work. For example, smaller orders at slightly less competitive prices or more recently submitted might be filled instead of a large order that is the longest standing at the most competitive price.

We model these priority rules as follows, from highest to lowest:

1) Most competitive price (p_i is maximal)

2) Longest standing (t_i is minimal)

3) Best "fill", measured by the percentage of shares filled of the larger of the two orders ($\frac{|q_i - q_j|}{\max(q_i, q_j)}$ is maximal).

We do not consider a formal mechanism for proving the time priority of an order correct, in part because we see no benefit in encrypting the timestamp of an order: orders are posted when they arrive, and that reveals the time they were posted. Further, this information is not readily exploitable.

We assume a bulletin board that orders are posted to; the market operator is required to accept new orders by adding them to the history H as soon as they arrive. We also assume that at the beginning of each new trading session the public, encrypted order book $\hat{B}$ has been verified by tracing through the previous day's history in H.

8.3.1 Assumptions

Our protocol rests on certain realistic assumptions. The operator and all traders possess the means for generating secure digital signatures. A universal, tamper-resistant clock must be accessible by all parties, such as that maintained by the US NIST, to preserve the integrity of timestamps. To prevent the operator from improperly failing to disclose instructions, there is a universally accessible bulletin board—not maintained by the operator—that records all activities of all parties and publishes them for anyone to see.[6] (All private data remain secure by encryption.) We assume the hardness of the composite residuosity problem supporting Paillier's homomorphic encryption scheme [114]. We assume that a computer network may be monitored for activity, and that even large amounts of activity can be examined for any information "leakage".

8.3.2 Encryption Method

We employ the homomorphic encryption scheme described by Pascal Paillier [114] and extensions published by Damgård and Jurik [51], Parkes et al. [115], and a use of Boudot's efficient range proofs [30]. We write the encryption of a value m with the market operator's public key and random help value r as $E(m, r)$. The properties of this cryptosystem allow construction of mathematical proofs of certain facts over the ciphertexts. For example, given only $E(m_1, r_1)$ and $E(m_2, r_2)$, one can prove a value

[6] We assume a bulletin board strictly separate from the operator so that traders' orders may be presumed received and posted on time without respect to their content. Because the operator can decrypt incoming orders, it is important that all incoming orders be posted by a neutral third party to require the operator to prove its actions are correct; a corrupt operator could delay or ignore incoming orders to benefit favored traders.

is within a constant range, e.g. $m_1 < n/2$; inequalities, e.g. $m_1 > m_2$; or generate new ciphertexts that are the sum of others, e.g. $E(m_1 + m_2, r_1 \cdot r_2) = E(m_1, r_1) \cdot E(m_2, r_2)$. We require these primitives for proving the correct operation of the market.

8.3.3 Processing Incoming Orders

Before orders arrive in any trading session, we recall that we assume the operator has proven the public, encrypted order book $\hat{B}$ correct by reference to the orders posted on the bulletin board in previous sessions. This means that all transactions may be performed with respect to existing orders in the order book without need for further proofs of their correctness or rank in the order book.

Limit Orders.

Any trader in our model may place a limit order according to the following protocol. Each limit ask order a_i is given a unique id i by the bulletin board and enters the market in the following manner. Note that the same method applies for bid orders b_i by interchanging "ask" and "bid" and reversing inequalities ($<$ becomes $>$).

Step 1. The trader encrypts the price p and quantity q and sends $(E(p, r_p), E(q, r_q), \mathtt{a})$ to the bulletin board. The bulletin board creates a unique identifier i, adds a timestamp t_i based on the current clock, publishes $\hat{a}_i = (E(p_i, r_{p_i}), E(q_i, r_{q_i}), t_i, \mathtt{a})$, computes the digital signature $SIGN_{BB}(\hat{a}_i)$ and both publishes it and sends it to the trader as a receipt. Only the operator can see what the p_i and q_i are.

Step 2. The trader privately sends the random help values r_{p_i}, r_{q_i} to the operator.[7]

Step 3. The operator privately decrypts the values in $\hat{a}_i$ to compute $a_i = (p_i, q_i, t_i, \mathsf{a})$, and verifies that the random help values correspond to the ciphertexts provided.

Step 4. The operator logs in H that order a_i was received at time t_i.

Step 5. The operator compares p_i to the best ask price, $p_{o(\mathsf{a},0)}$ and the best bid price, $p_{o(\mathsf{b},0)}$ and proceeds in one of four ways:

- If the incoming ask order is priced at less than or equal to the highest priority bid, i.e. $p_i \leq p_{o(\mathsf{b},0)}$, the operator matches a_i with all outstanding bid orders whose prices are $\geq p_i$ up to the quantity q_i in order of priority. If there are not enough to fill a_i, it becomes the most competitive ask order on the order book afterward.
- If the incoming ask order is priced between the highest bid and the lowest ask price, i.e. $p_{o(\mathsf{b},0)} < p_i < p_{o(\mathsf{a},0)}$, the operator adds it to the order book.
- If the incoming ask order is priced equal to the lowest ask price, i.e. $p_i = p_{o(\mathsf{a},0)}$, the operator adds it to the order book.
- If the incoming ask order is priced higher than the lowest ask price, i.e. $p_i > p_{o(\mathsf{a},0)}$, the operator adds it to the order book.

[7]This is required to prevent other traders from exploiting the malleability of the homomorphic encryption scheme to submit bids based on a function of another trader's bid, e.g. "his bid plus 10 cents." Knowing the random help value implies knowing the decryption, so provided the cryptosystem is secure and the random help values are secret, no trader can submit a correct random help value for a ciphertext based on another trader's encrypted values.

Step 6. The operator updates H on the bulletin board with the details of any trade that resulted from receiving a_i.

Step 7. The operator recomputes the ordering function $o(s, \mathit{rank})$ such that the rank of all orders in B is defined and correct.

Step 8. The operator updates its private B and publishes $\hat{B}$ on the bulletin board with the new set of encrypted orders.

Step 9. The operator issues proofs of correctness of its actions on the bulletin board. Specifically, it proves the necessary inequalities to pigeonhole the incoming limit order a_i in its proper priority ordering, maintaining the invariant that the all outstanding orders in B and $\hat{B}$ are ordered according to priority.

Step 10. Anyone who wishes may verify the operator's public proofs.

Market Orders.

A trader in our model may also place a market ask order a_i (or bid b_i). The protocol differs from the limit order protocol given above only in Step 5:

Step 6. The operator matches the incoming market ask order a_i with the k highest priority bid orders $b_{o(\mathtt{b},0,\ldots,k)}$ such that the $k-1$ highest bids do not fill a_i but k do, and executes the trade(s) on all matched orders.

Executing Trades on Matched Orders.

The operator must prove that the quantity of the k multiple limit orders a large order is matched with is greater than or equal to the quantity of the market order,

and that the sum of the quantities of the most competitive $k-1$ limit orders is strictly less than the quantity of the market order.

Two orders a_i and b_j are matched when the bid price meets or exceeds the ask price, i.e. $p_j \geq p_i$. If the quantities are equal, $q_i = q_j$, the trade is executed and both orders are removed from the order books B and $\hat{B}$ and the transaction is logged in the history H. Formally, to log the transaction the operator adds a journal entry to H $h_{i,j} = (\hat{a}_i, \hat{b}_j, t_{i,j})$ with its signature $SIGN_{MO}(h_{i,j})$. The time $t_{i,j}$ is the time reported by the universal clock at the time the order was executed. The operator also posts the following proofs on the bulletin board:

- A proof that $p_j \geq p_i$ given $E(p_i, r_{p_i})$ and $E(p_j, r_{p_j})$.
- A proof that $q_j = q_i$ given $E(q_i, r_{q_i})$ and $E(q_j, r_{q_j})$.

If the quantities differ, the order for fewer shares is fully filled and the order for more shares is partially filled. Then, the smaller order (w.l.o.g. a_i) is removed and the larger order (w.l.o.g.) b_j's quantity is updated in the order books B and $\hat{B}$. Formally, the entry b_j in B is replaced with $b_j = (p_j, (q_j - q_i), t_j, \mathtt{b})$, and in $\hat{B}$ with $\hat{b}_j = (E(p_j, r_{p_j}), E(q_j, r_{q_j})/E(q_i, r_{q_i}), t_j, \mathtt{b})$. Anyone can verify the correctness of the new published $\hat{b}_j$ by computing the quotient of the previously published encrypted values $E(q_j, r_{q_j})$ and $E(q_i, r_{q_i})$, which is known to be an encryption of their difference. The transaction is logged in the history H as above with a similar journal entry $h_{i,j} = (\hat{a}_i, \hat{b}_j, t_{i,j})$ and signature $SIGN_{MO}(h_{i,j})$. The operator also posts the following proofs on the bulletin board:

- A proof that $p_j \geq p_i$ given $E(p_i, r_{p_i})$ and $E(p_j, r_{p_j})$.

- A proof that $q_j > q_i$ given $E(q_i, r_{q_i})$ and $E(q_j, r_{q_j})$. This is done by showing that $(E(q_j, r_{p_j})/E(p_i, r_{p_i})) \cdot E(-1, 1)$ is the encryption of a value $(q_j - q_i - 1) < n/2$. (This proves that no wraparound occurred; we subtract 1 from $q_j - q_i$ to prove a strict inequality.)

One minor issue in a market without transparent prices is that a limit order may be submitted to the market that is more competitive than it needs to be to clear. For example, a trader might post a new limit order to sell at $20.05 when there is a standing order to buy at $20.09. In transparent markets, this would obviously never happen except in cases of error. Choosing the clearing price for such situations is a matter of market design. With the primitives we have described, it is possible to prove correct a clearing price based on the standing order's price, the incoming order's price, the mean of the two (within one tick), or indeed any linear function of the two prices, without revealing the price itself or any information not implied.

Once two orders are matched and the proofs posted, a clearing agent will be responsible for transferring the ownership of the shares at the correct settlement price. The market operator will send the clearing agent the random help values necessary to verify the correctness of the execution price and number of shares from the history posted on the bulletin board. The agent then verifies the trade and settles it.

In addition to sending information to the clearing agent, any information published about the state of the market is proven at this point on the bulletin board. For example, the auctioneer might reveal the random help values associated with the determined clearing price and matched quantity to provide "last trade" tick data, or update proofs of market depth, bid/ask prices, etc. Typically the "market price" of

a security for any period is the price at which it was last traded during that period; thus, publishing provably correct market prices is straightforward.

8.3.4 Post-Trade Reporting

The market operator can report clearing prices by revealing the random help values of the encrypted orders in the history H after any specified delay. Immediate revelation may be a problem in the event a partial fill is revealed and the remainder is still on the market: its price is now public. Facts similar to those provable for limit orders may be proven about trades after the fact, for example, volume, average price, closing price, etc. Post-trade transparency is as easily controlled by market designers as transparency during other phases of market activity, and we leave the question of appropriate reporting rules open for this reason.

8.3.5 Adversaries and Attacks

The adversary we are most concerned about in this work is the unethical or parasitic trader who exploits (presently public) market information for profit in a way that discourages placement of limit orders. A secondary class of adversary is a dishonest market operator who may attempt to profit by exploiting the now private market information via trading or disclosure for compensation. We do not consider as adversaries parties with private information external to the market's operation, such as employees with proprietary information about traded companies.

Traders

We first consider attacks by parties who do not possess any insider access to the market operator or its systems. These traders may either attempt to circumvent the cryptographic security of the system or exploit the information provided in new ways. Provided cryptographic keys of adequate security are chosen to prevent a brute-force attack, cracking the encryption scheme itself is believed to be intractable under the Decisional Composite Residuosity Assumption described in Paillier's work [114].

The semantic security of the probabilistic Paillier cryptosystem protects the encrypted values against chosen plaintext attack. (For example, using a deterministic encryption of prices would be insecure, because an adversary could try all realistic prices and identify the values.) Paillier's scheme is not secure against an adaptive chosen ciphertext attack; indeed, the malleability of the scheme that enables the homomorphic properties we employ implies this insecurity. However, mounting a successful chosen ciphertext attack against our protocol does not seem a significant threat, as the only way a value can be decrypted is in the event someone is willing to trade it. Thus, any party attempting to gain information by submitting a chosen ciphertext as information must also be willing to execute any trades based on that information.

We have not identified any additional parasitic trading practices that could be employed using a cryptographic securities exchange. Since we are not adding any information into the marketplace – only allowing designers to restrict information – we believe that there are no new exploits that would not be possible in an ordinary market with an open limit book.

This said, we reiterate that some parties may attempt to gain information from

the marketplace by placing orders. For example, one could discover the price for the most competitive ask order by placing an order to buy one share at the market. Alternatively, a trader might place limit orders at various prices to see where they fit into the order book, in order to gain information about the price points, and then retract them. However, no trader may observe anything about the market without fundamentally changing the market: a "probe" share purchased revealed the price *for that share only*, and afterward, the number of shares at that price remains unknown and becomes smaller; probe limit orders enter the market and always bear the risk of being executed.

Several solutions to this problem come to mind. First, at a significant but tractable complexity cost, the marketplace could maintain not a strict ordering over all orders, but a partial ordering in which only the minimum information required to prove correctness is revealed. Thus incoming orders that were not competitive (and likely to be filled) would be proven only to be less competitive than the most competitive order. This would significantly limit the ability of a trader to count trades above a particular price by placing limit orders. Second, the market operator or market makers could place random numbers of zero-quantity limit orders on the marketplace so that there would be a large number of orders at every price point. Third, market designers could limit such exploitative practices by limiting order frequency, sizes, or specifying a minimum duration on the market.

Finally, for ultimate security, the market operator can employ a mix network (see the discussion in Section 7.2.1) to mask all of the remaining orders after each transaction, discarding canceled orders and adding new limit orders and empty orders

for zero shares to mask the number of orders in the market. Each cleared order would be proven correct by proving price ordering across the entire market, then proving which trades should be executed (if any). Such an approach would prevent observers from tracking the position or number of existing orders to glean extra information from the market. Of course, such a protocol would be significantly more expensive from a computational perspective, but given that large block trade commissions can exceed $1,000 per trade on some ATS's, it is certainly economically feasible to provide such a service.

The Market Operator

A more insidious attack is if a dishonest market operator, possibly in collusion with another trader, exploits its valuable private information or gives preference to particular traders. We recall our assumption of a bulletin board operated by a third party to prevent the market operator from discarding dispreferred orders, or delaying their publication until after preferred orders are listed. With this, an unscrupulous market operator cannot issue valid proofs of correctness of matched trades, but he could still selectively reveal information to preferred traders. We reiterate that despite this implied trust in the market operator, our architecture provides for two improvements over existing markets: information can be specifically controlled and is possessed by only one party (instead of the entire market), and the market operator may not manipulate the market by front-running or matching orders on any basis other than the published rules.

That said, the partial trust of the operator is a strong assumption, and solutions

to enhance that trust merit discussion. One answer is to distribute the trust in the market operator among a group of parties, similar to the approach Bogetoft et al. describe [26]. This may be challenging from a business perspective but nonetheless possible. Another solution involves careful network, hardware and software security, employing special purpose hardware (e.g. that used in Trusted Computing architectures) that only runs software approved and signed by a third party, employing a trustworthy and auditable source of random data, and monitoring all network traffic to detect any communications that might leak information.

8.4 Example Order Book and Transactions

This section describes incoming orders and how trades are identified and executed. Table 8.1 shows a sample order book B. The public, encrypted order book $\hat{B}$ is equivalent, except that the quantities and prices are encrypted. **R** indicates *rank*. Orders are always ranked in priority order. Each order's rank is defined according to the priority rules outlined above (best price, oldest) and randomly selected in the case of a tie.

We first consider an incoming market order to purchase 700 shares of the stock. The trader constructs $\hat{b} = (\_, E(700), \_, \texttt{b})$ and posts it on the bulletin board. The bulletin board assigns ID $i = 25$ and timestamp $t_i =$ 09:44:32 and publishes $\hat{b}_i = (\_, E(700), t_i, \texttt{M})$. For clarity, we will use i for the ID of each incoming order in the following text to more clearly distinguish it from the limit orders.

The market operator sees the market order on the bulletin board, decrypts $\hat{b}_i$ to $b_i = (\_, 700, t_i, \texttt{b})$, and matches two trades ($a_1 4$, $a_1 2$) to fill the order. It adds journal

entries to the history H and publishes proofs on the bulletin board:

- $H \leftarrow h_i = \hat{b}_i$
- $H \leftarrow h_{14,i} = (\hat{a}_{14}, \hat{b}_i, t_{14,i} = 09{:}44{:}33)$
- $H \leftarrow h_{12,i} = (\hat{a}_{12}, \hat{b}_i, t_{12,i} = 09{:}44{:}33)$
- Quantities: $q_{14} + q_{12} \geq q_i$ and $q_{14} < q_i$
- Priority: $q_{12} < q_{13}$

The operator then updates B (and $\hat{B}$) by removing order a_{14} and updating $q'_{12} = 300 - (700 - 600) = 200$ (and $\hat{q}'_{12} = E(q_{12})/(E(q_i)/E(q_{14}))$). Anyone can verify that the updated encrypted quantity $\hat{q}'_{12}$ is correct by comparing it with functions of the quantities of the other orders.

In a second example, a trader posts a new limit ask order $\hat{a} = (E(\$20.03, E(1200), \_, \mathtt{a})$ to which the bulletin board assigns $i = 15, t_i = 09{:}46{:}02$. The market operator sees it, decrypts it, and concludes it is more competitive than the most competitive bid. He adds journal entries to H, removes $b_{o(\mathtt{b},0)}$, matches a_i with b_{22} and adds the remainder a'_i to B and $\hat{a}'_i$ to $\hat{B}$, preserving the priority order invariant, and publishes:

- $H \leftarrow h_i = \hat{a}_i$
- $H \leftarrow h_{i,22} = (\hat{a}_i, \hat{b}_{22}, t_{i,22} = 09{:}46{:}04)$
- Proof of correct quantities: $q_i > q_{22}$
- Proof of clearing price (as required) and price position: $p_i \leq p_{22}$, $p_i > p_{24}$

R	**ID**	**Time**	**Qty**	**Ask**
3	11	09:34:42	2500	$20.13
2	13	09:39:23	500	$20.10
1	12	09:39:23	300	$20.10
0	14	09:41:06	600	$20.09
R	**ID**	**Time**	**Qty**	**Bid**
0	22	09:37:14	1000	$20.05
1	24	09:43:42	500	$20.02
2	23	09:41:23	800	$20.00
3	21	09:30:06	1700	$19.96

Table 8.1: Order Book B_1

R	**ID**	**Time**	**Qty**	**Ask**
3	11	09:34:42	2500	$20.13
2	13	09:39:23	500	$20.10
1	12	09:39:23	200	$20.10
0	15	09:46:02	200	$20.03
R	**ID**	**Time**	**Qty**	**Bid**
0	24	09:43:42	500	$20.02
1	23	09:41:23	800	$20.01
2	26	09:50:33	200	$19.98
3	21	09:30:06	1700	$19.96

Table 8.2: Order Book B_4

In a final example, a trader posts a limit bid order $\hat{b} = (E(\$19.98, E(400), \_, \mathtt{b})$ to which the bulletin board assigns $i = 26, t_i = 09{:}50{:}33$. The market operator sees it, decrypts it, and places it in the order book in the appropriate position It adds a journal entry to H, adds the order b_i to B and $\hat{b}_i$ to $\hat{B}$, preserving the priority order invariant, and publishes $H \leftarrow h_i = \hat{b}_i$ and the proofs of priority $p_i < p_{23}$ and $p_i > p_{21}$. The order book is now as shown in Table 8.2.

8.5 Conclusions and Future Work

Clearly, providing controllable transparency of market information in securities exchanges together with proofs of correctness (both of information and of the market operation) is an important application of homomorphic cryptography. The protocol presented here is simple to understand, closely related to existing financial market protocols, and does not rely complex cryptographic primitives that might discourage its use among traders. Finance research has already started to study the implications of different levels of partial transparency, seeking to ensure liquidity and limit exploitation. Cryptography can be used to prove correct operation according to specified rules even under partial transparency.

We envision a broad range of future work based on the protocol we have presented and similar ideas. For instance, market designers might want support for more expressive order types, such as fill-or-kill, immediate-or-cancel, order-cancels-order, or stop orders maintained by the market. Our protocol could also easily be extended to open call auctions or periodic clearing models (such as POSIT). The market operator might wish to prove a less revealing ordering of the limit orders in the order book.

Support for other specialists and liquidity providers' functions could be added by selective revelation.

Other more creative exchanges are possible in our setting. For example, integrating other ECN's with a cryptographic securities exchange may be of particular use in bridging the gap between block trades and ordinary securities trading. Cryptographic derivative markets for options and indices whose prices are tied to the activity in underlying securities' order books are another important possible extension of our work. In the next chapter, we explore models of cryptographic combinatorial securities exchanges, where entire baskets of securities may be exchanged, rather than blocks of a single security.

We have conducted an initial empirical analysis of the computation cost for running such a system, and arrived at a conservatively high estimate of 5 cents (US) to place and verify an order. Our experiments used a low end, dual Pentium IBM x-server with no special cryptographic hardware. This is inexpensive enough to be feasible in practice, although we leave a full efficiency analysis, perhaps in conjunction with a prototype, to future work.

Chapter 9

Cryptographic Combinatorial Securities Exchanges

We present a useful new mechanism that facilitates the atomic exchange of large baskets of securities in a combinatorial exchange. Cryptography prevents information about the securities in the baskets from being exploited, enhancing trust. Our exchange offers institutions who wish to trade large positions a new alternative to existing methods of block trading: they can reduce transaction costs by taking advantage of other institutions' available liquidity, while third party liquidity providers guarantee execution—preserving their desired portfolio composition at all times. In our exchange, institutions submit encrypted orders which are crossed, leaving a "remainder". The exchange proves facts about the portfolio risk of this remainder to third party liquidity providers without revealing the securities in the remainder, the knowledge of which could also be exploited. The third parties learn either (depending on the setting) the portfolio risk parameters of the remainder itself, or how their own

portfolio risk would change if they were to incorporate the remainder into a portfolio they submit. They submit bids on the commission, and the winner supplies necessary liquidity for the entire exchange to clear. This guaranteed clearing, coupled with external price discovery from the primary markets for the securities, eliminates the difficult combinatorial optimization problem. This latter method of proving how taking on the remainder would *change* risk parameters of one's own portfolio, without revealing the remainder's contents or its risk parameters, is a useful protocol of independent interest.

9.1 Introduction

In Chapter 8 we introduced the idea of a cryptographic securities exchange for individual equities, motivated by the unfavorable price impact and possible exploitation of information associated with block trades.[1] In that chapter, we consider an exchange of single securities, and, typically, securities are traded as single asset types in most alternative trading systems.

In this chapter, we consider a cryptographic combinatorial securities exchange, where entire *baskets* of securities may be bought or sold, rather than single positions. This has important applications for portfolios of securities where entering the various positions in the portfolio piecemeal would subject the investor to increased portfolio risk. After all, if a large portfolio is optimized to have certain correlations among its assets, and it takes hours to find a counterparty to fill various positions in the portfolio, the orders filled first will have a different risk profile than the intended

[1] Exchanges of very large positions of securities.

portfolio.

For example, say a particular investor believes that Toyota will outperform Ford, but does not wish to undertake any risk from general market or automotive sector movements. He takes a long position in Toyota and a short position in Ford of the same size. If the automotive sector or the general markets move up, then the gains in Toyota will offset the losses in Ford. The investor's problem is that he needs to enter these trades at *exactly the same time*. If he is taking a large position in this portfolio, then he would be exposed to risk in general market and automotive sector movements between the time he closed the first and second position. Our exchange, which provides for atomic trades that are guaranteed to clear, eliminates such risks.

9.1.1 Existing Commercial Protocols

In Chapter 8, we pointed out a few problems with the existing alternative trading systems (ATS's) for block trades. Institutions still fear that knowledge of their liquidity can be exploited in various ways, and rely on information brokers like Liquidnet who strictly limit membership to the trading network to parties who are only trading for liquidity reason. A second problem is that there is typically no guarantee of execution. Finally, there is no mechanism for trading an entire basket at once, eliminating portfolio risk while the execution of the trades is going on.

We work to ameliorate all of these concerns: our proposal enhances trust by not only keeping trades secret until the market is to clear but also proving the results correct; it also improves liquidity by giving the our exchange an efficient mechanism to guarantee execution for all of the trades submitted to it— while still keeping the

particular equities in the incoming institutions' baskets secret; and it provides an atomic basket trading paradigm.

Currently for large basket trades (involving more than one security), the transactions are too complex for the pairwise trade matching that existing ATS's like Liquidnet and Pipeline offer. Institutions who need to trade a basket of securities atomically to maintain the integrity of a diversified portfolio are not willing to undertake the risk of executing the trades one security at a time. Thus, institutional investors who wish to trade several large positions at once in a *basket order* typically hire an investment bank. They describe the basket to a small number of trusted investment banks who agree to provide liquidity, without disclosing the exact securities that comprise the basket in advance—information that could be exploited. When deciding how much to charge for liquidating a basket, the banks learn only certain risk parameters, such as index membership, daily trading volume, and market correlation; these enable them to estimate their risk and costs in the absence of complete data.

Our new cryptographic combinatorial exchange provides the improved efficiency of institution-to-institution trading with the reduced portfolio risk from guaranteed execution of atomic basket trades. Cryptography makes such an exchange feasible by providing necessary trust: exploitable data remain secret, and every action and result can be proven correct.

In our combinatorial exchange, institutions submit baskets of buy and sell orders which are filled by other institutions' sell and buy orders (respectively). The unfilled orders comprise a remainder basket, which clears the exchange when filled by a cooperating third party (assumed to be an investment bank). Prices for each security are

determined by the primary markets, so that the exchange need only discover trading interest.

Because direct disclosure of the remainder would permit exploitation of that information, institutions submit their baskets in an encrypted form which can then be used to derive an encrypted remainder. Then, the exchange can prove facts about this encrypted remainder to the investment banks without revealing its contents. Moreover, we describe how to construct a proof of how a bank's risk on a portfolio changes by taking on the remainder, by using encrypted forms of the remainder and that portfolio. This enables the banks to accurately estimate commissions to charge the exchange for providing the necessary liquidity.

The guarantee of order execution makes this market extremely attractive. The institution need not wait for another trader to indicate interest: if there is opposite interest, it will be used; if there is not, then the bank provides the liquidity. This offers an unprecedented market efficiency: the exchange offers a mix-and-match of cheap institution-to-institution liquidity wherever possible, only using the more expensive bank-provided liquidity where necessary. It also means that institutions can count on their order being filled completely in a reasonable amount of time, eliminating portfolio risk from partial fills and reducing the risk of holding securities while trying to trade them.

Another advantage of guaranteed order execution is that it prevents exploitation of even dark market centers by orders designed to search for liquidity; traders do not need to show their hand by entering an order into the dark pool because they are guaranteed their order will be filled.

This approach may also be more compatible with recent securities regulations. In the United States, the National Best Bid and Offer (NBBO) system and National Market System (regulated by the so-called "Reg NMS") govern the prices at which publicly traded securities may be exchanged; Europe recently adopted a new Markets in Financial Instruments Directive (MiFID) that has a similar mandate. Regulatory compliance is a significant challenge for block trading systems that wish to hide data, and it may be difficult to legally operate an exchange in which standing limit orders are meant to be kept secret from any national market. Our model, which only discovers liquidity and derives prices from the primary markets, should be more compatible with ever-tighter regulation.

We believe this to be the first characterization of a cryptographic combinatorial exchange: a number of participants submit bundles to buy and sell goods (in our example, securities), and the market finds an optimal allocation of trades to maximize the benefit of all participants. While such combinatorial exchanges typically require significant computation to find optimal allocations,[2] our exchange makes two important simplifications that eliminate the hard combinatorial problem. First, prices are defined externally by the primary markets, and second, our clearing of the remainder via a third party means that all bundles are filled and the market clears at equilibrium.

9.1.2 Related Work

Szydlo [148] first proposed the application of zero-knowledge proofs to disclosing facts about equities portfolios. In his highly relevant and pioneering work, a hedge

[2]Indeed, even defining "optimal" in such an exchange is challenging!

fund proves that its portfolio complies with its published risk guidelines without revealing the contents of its portfolio. Szydlo's proofs are not situated in a transactional context, but rather in the context of a hedge fund reporting portfolio risk characteristics that are based on the claimed securities in the portfolio. In our case, we are interested in proving portfolio risk in order to liquidate a newly derived remainder basket computed from a combination of many incoming baskets, not a private portfolio that will never be revealed.

Another difference in our work is the use of encryption over commitments. Encryptions allow the exchange to issue proofs about combinations of the institutions' baskets without requiring their continued involvement. Were we to employ commitments, we would require institutions to decommit their baskets before computing the remainder; this provides an opportunity for repudiation. While the homomorphic Pedersen commitments Szydlo employs are more efficient than homomorphic encryptions, we desire nonrepudiation: once a basket is committed to in a transaction, the trader may not later refuse to reveal that basket. Since any non-repudiatable commitment is equivalent to an encryption,[3] we elect to employ encryptions directly. This may also mitigate so-called protocol completion incentive problems (see [31] for a related discussion in the context of auctions), because traders who lose their incentive to participate cannot benefit from refusing to complete the protocol.

Surprisingly little academic research has been published on applications of cryptography in securities trading; we refer the reader to our discussion in Section 8.1 for a brief survey of relevant recent work.

[3]To enjoy nonrepudiation, a commitment must be deterministically invertible. A function that is binding, hiding, and invertible (presumably via some secret) is clearly equivalent to an encryption.

More work has been done on combinatorial exchanges, in which buyers and sellers come together in a common exchange to trade bundles of various goods (where bundles may have instructions to buy or sell, or both.) In the general case, solving the price and winner determination problems in a combinatorial exchange is extremely difficult; in our cryptographic combinatorial securities exchange, we get around these by taking all prices from the fair prices already established by the primary markets (price determination), and employing "liquidity providers" who guarantee enough liquidity for the entire exchange to clear (winner determination). See Parkes et al. [117], and Smith et al. [143] for a formal treatment of combinatorial exchanges and related work. Parkes et al. [116] have also implemented an "iterative combinatorial exchange", an interesting new mechanism that provides for more efficient price and winner determination.

9.2 Cryptographic Combinatorial Securities Exchanges

Our cryptographic combinatorial securities exchange offers basket traders guaranteed execution and efficient liquidity discovery. It keeps information completely secret until it is necessary, eliminating opportunities for fraud, and proves every result correct without revealing unnecessary information.

Our protocol is simple: institutions submit encrypted baskets; the exchange closes; the exchange creates an encrypted remainder and proves risk characteristics to third party liquidity providers; these investment banks bid on their commission; and the

winning bank clears the market by liquidating the remainder. Prices clear at prices determined by the primary markets.

The basic cryptographic protocols supporting provably correct, secrecy-preserving computation over private inputs described in Chapters 2 and 3 are sufficient to construct our exchange. Our protocol does not depend on specific features other than those therein, we do not burden our exposition with specific implementation details. Rather, we assume implementors of our protocol will select an underlying cryptosystem appropriate to their specific needs at the time. Moreover, these protocols are practically efficient and support the calculations of risk and interval proofs essential to our protocol. We discuss the implications of the partial trust in a third party and mechanisms for mitigating such trust in Section 9.6.

9.2.1 The Protocol

We consider n trading parties P_i, where $i \in [1, n]$, each of which submits a basket B_i, comprised of m securities S_j, where $j \in [1, m]$. Thus in a universe of 6 securities, B_3, P_3's basket, might be $\langle 0, -20000, 32000, 0, 45000, 0 \rangle$. The double subscript notation B_{ij} denotes the (unencrypted) quantity of security j in P_i's basket; in our example, $B_{35} = 45000$. $E(B_{ij})$ is the encrypted form of one such value. Zeroes are included to hide the number of distinct equities in the basket, though a fixed basket size simplifies future computations.

Since most underlying cryptosystems employ modular arithmetic, short positions can be easily represented as "negative numbers" (that is, very large numbers that are the additive inverses of the corresponding positive number).

An encryption of a basket of equities is simply a set of quantities, one for each equity in the universe, including zeros. For visual comfort, we may write $E(B_i)$ as the encryption of an entire basket, which is in fact m separate encryptions: $\langle E(B_{i1}), E(B_{i2}), \ldots, E(B_{im})\rangle$.

Step 1. The market operator announces clearing times, the universe of equities to be traded on the exchange, and any rules governing the composition of baskets participating in the exchange. If time-lapse cryptography (TLC) [129] or a similar technique used to enforce nonrepudiation requires posting of public information (for example, a public TLC encryption key), the market operator posts that.

Step 2. Before each clearing time, each trader P_i chooses which equities she wishes to trade and creates basket B_i and encrypted form $E(B_i)$. Each then creates a commitment to her basket, $\mathbf{Com}_i(E(B_i))$, and publishes that commitment where the exchange and other parties to the transaction can see them.

Step 3. When the clearing time is reached, the traders decommit: they publish encryptions of their baskets and any proof necessary to prove their prior commitments were valid. (If a trader fails to decommit, and a nonrepudiation technique is used, the commitment is forced open and the encryption of his basket is published.)

Step 4. Either the market operator, or each individual P_i, proves, using the now public $E(B_i)$, that the basket P_i submitted conforms to any announced basket composition requirements. Because P_i encrypted the basket herself, she is capable of proving her basket meets any exchange requirements (see Section 9.5) without the cooperation of the market operator.

Step 5. Everyone can compute the "remainder" basket B_R by computing a func-

Security	B_1		B_2		B_3		B_4		R
ABC	+500		-200		0		0		+300
DEF	+300		-800		+300		+200		0
GHI	0	+	+100	+	-300	+	0	=	-200
JKL	+200		0		-400		+300		+100
MNO	-800		0		+500		0		-300

Table 9.1: Example set of cross-clearing portfolios with a "remainder"

tion on all of the encryptions of traders' baskets. Table 9.2.1 illustrates an example of this on unencrypted values, while using our standard notation, we write:

$$B_R = \langle \bigoplus_{i=1}^{n} E(B_{i1}), \ldots, \bigoplus_{i=1}^{n} E(B_{im}) \rangle \quad (9.1)$$

$$= \langle E(\sum_{i=1}^{n} B_{i1}), \ldots, E(\sum_{i=1}^{n} B_{im}) \rangle \quad (9.2)$$

Step 6. The market operator privately decrypts the baskets, and obtains the unencrypted remainder basket.

Step 7. The market operator proves facts about the composition of the remainder basket B_R to the third party liquidity providers, who individually or jointly determine transaction costs for the remainder basket and agree to provide liquidity to the pool.

Step 8. After the market-clearing liquidity has been secured, the market operator announces the protocol is complete, and issues each institution a proof of its share of the commission based on the other encrypted baskets.[4] The market clears at prices fixed in accordance with a published standard procedure. For example, the market might clear at the midpoint between the bid and ask quoted on the current primary

[4]Our work concerns itself with the privacy of trade information, not the identity of the traders. Other cryptographic protocols may be employed to provide any required anonymity.

market, or an agreement to trade at the volume-weighted average price for a particular period of time. The mechanics of clearing securities trades are beyond the scope of this work; we assume that all parties trade with a trusted intermediary who accepts all long positions and distributes all short positions, clearing the market.

9.3 Secrecy-Preserving Proofs of Impact on Portfolio Risk

In the introduction, we describe how large basket orders are traded by revealing portfolio risk measurements of the baskets themselves, rather than the actual risk undertaken by the banks accepting the baskets. We propose a secure system that makes price discovery for basket trades more accurate by offering banks limited but more specific characteristics of their actual risks — how the risk of their inventory changes — not the characteristics of the incoming basket.

Our system acts as a partially trusted third party, accepting encrypted forms of the institution's portfolio and the bank's book, and providing a set of risk characteristics of the bank's resulting book after the integration of the equities in the portfolio. The system proves these characteristics correct in a zero-knowledge fashion based on the encrypted inputs, to assure the bank that it received an accurate picture even if it does not win the bid. (Presently, only winners can verify the correctness of the submitted values because they are the only party who ever discovers the actual contents of the basket.)

Finally, we remark that wherever we refer to a bank's "inventory", the bank may

submit any representative portfolio to the system and compute the risk of accepting the basket on the basis of risk changes in this particular portfolio. This may be due to reluctance to reveal the exact portfolio to even a partially trusted third party, or to achieve improved price discovery by a specially tailored portfolio.

9.3.1 Mechanics of the Protocol

The protocol is comprised of a series of simple steps: the parameters of the transaction are agreed on; the two transacting parties publish their encrypted information to all; the two parties send secret information to the partially trusted third party; the third party issues proofs to one party about the portfolio risk; and that party verifies the proofs using the published information.

Step 1. The institution and bank agree on a set of risk characteristics to evaluate in the resulting book. This step protects the secrecy of the institution's information while providing enough information to the bank to quote an accurate price. The institution may also require that certain outputs be reported as "bounds", where the results are only quoted accurately enough for the bank to price the portfolio by proving they lie within a certain small range. This is of extreme importance to prevent the banks from "backing out" private information from the encrypted data by carefully constructed queries. See also the more detailed discussion in 9.3.2.

Step 2. The institution prepares a list of triples:[5]

□ Identifying code (ticker, CUSIP, etc.)

[5]The dark square ■ indicates an encrypted private value; the open square □ indicates the information is unencrypted.

- Number of shares
- Value quotation (e.g. previous close × shares)

The encryptions are carried out in accordance with the underlying cryptographic protocol.[6]

The institution shares these encrypted data with the system and the banks. For complete security, the basket should include all equities in the trading universe, with encrypted zero values for quantity and total value being used for any equities not in the portfolio. Short positions are naturally represented by negative values. This also allows the institution to keep the total number of equities in the basket secret if desired.

Step 3. Each bank prepares a similar data set of triples for its inventory, into which the basket would be integrated, and shares this encrypted portfolio with the system. It does *not* need to share it with the institution.

Step 4. For each bank, the system privately decrypts the institution's portfolio and the bank's inventory and computes the resulting portfolio and its risk characteristics. It then creates a list of statistics about the resulting portfolio and proofs of their correctness as described below in Section 9.3.3. It reveals these proofs to each bank, which in turn verifies that they were computed correctly using its encrypted portfolio and the encrypted portfolio provided by the institution.

Step 5. The bank examines the new risk characteristics of the resulting portfolio, estimates carrying and execution costs and submits a bid to the institution. (In practice, the computed characteristics might be sent to a portfolio management software

[6]Providing the value quotation is a matter of convenience, as the encrypted value can be computed as the encrypted product of public previous close price and the encrypted number of shares.

system that compares the "before" and "after" portfolios to automatically estimate risk and hedging costs.)

9.3.2 What Information Should Be Revealed?

Presently, institutions submit the characteristics of their baskets to banks in spreadsheets with specific numbers in each category. This process "leaks" information, especially where the number of equities in a particular category is small. Occasionally, the information can create obvious implications: for example, if there is only one equity listed in the telecommunications sector, comprising 89,000 shares whose total value is $3,546,650, the bank probably has an excellent idea of the company's name. Institutions sometimes "white out" some information in their basket descriptions to prevent such information leakage, usually to eliminate obvious information leaks.

Yet even when such information is redacted, rigorous statistical analyses of the information submitted can still yield information about the composition of the baskets, and this is also possible in more complex situations where a large number of equities contribute to one line-item. Since values are often supplied to the penny, if the number of equities, total dollar amount as of a particular market close, and total number of shares is known, it is possible that a computer could efficiently search the possible baskets created by equities in that sector and propose a small number of alternatives to the bank. While we have no reason to believe that the reports are being so exploited by the banks, eliminating any potential information leakage while still providing accurate risk assessments is an important benefit of our proposed system.

Because the cryptographic framework we describe supports interval proofs on en-

crypted values (or functions on encrypted values) the system can reveal approximate risk characteristics that are sufficient for price discovery but are more resistant to statistical analysis to back out the composition of the baskets. For instance, instead of reporting the sector breakdown exactly, the system can report values rounded to the nearest percentage point or thousands of dollars or shares. Although there is no reason that institutions can't submit baskets with such obfuscated data, they would not be able to prove it correct without cryptography. The ability to reveal "just enough" information (while still proving it correct) is an important feature of our proposal.[7]

9.3.3 How the Information Is Revealed

Our protocol is a useful extension of Szydlo's work referred to in Section 9.1.2; but rather than proving portfolio risk of a single portfolio, we are interested in revealing facts about a hypothetical portfolio that results after a bank with a large inventory (which it wants to keep private) accepts a basket of equities (which the institution trading the basket wants to keep private.)

Once our protocol is followed, the system privately knows the combined portfolio, and the bank knows its own portfolio and the encrypted quantities of equities in the incoming basket. To reveal a fact, the system obtains the result of the desired computation and sends the result to the bank, along with special verification data that allow the bank to verify the result. The form of these data depends on the type of system employed; in this work we assume a general framework based on

[7]See Section 9.6 for a discussion of why this feature is best supported by protocols based on a partially trusted third party.

the cryptographic primitives described in Section 2.1.4. The bank then performs a computation using the hidden data provided by the institution and data from its own portfolio, and verifes the result of the computation using the special verification data provided by the system.

We recall from Section 2.1.4 the following primitive operations necessary to reveal the portfolio risk profile:

- Compute a polynomial function of multiple encrypted values, or encrypted values and constants
- Decide whether one encrypted value is greater than another
- Decide whether one encrypted value is (not) equal to another

9.3.4 Computing the Combined Portfolio

After **Step 3** above, the system now knows both the institution's incoming basket and the bank's inventory. It verifiably computes the new quantities for every equity in the universe by creating a "combined portfolio" with the combined quantity and value of each equity. (Again, we assume that short positions are represented by a negative number of shares and negative total value.)

Computing this verifiable, encrypted combined portfolio is simple; in many cases, the banks can also compute the encrypted combined portfolio (but importantly learn nothing from it.) For each equity in the portfolio, the system computes the sum of the institution's and bank's number of shares and total value. It then publishes the new encrypted portfolio and proves to the bank that this encrypted portfolio is

equivalent to the sum of the bank's inventory and the encrypted institution's basket. We recall that both the baskets and the bank's inventory are represented as a list of triples: $\langle$Equity ID, E(shares), E(value). Using the cryptographic primitives above, the system can simply add up the number of shares and the values for each equity in the universe to obtain a new combined portfolio.

The encrypted values comprising this "combined portfolio" can now be used to prove facts about it in zero-knowledge as described in the following sections.

9.3.5 Portfolio Value and Dividends

In most cases, the incoming basket order will involve long and short trades, and an important element of the risk is the "skew" — the difference between the total value of the short and long trades. Sometimes, when an institution is trading a basket with a significant skew (or even entirely one-sided) it may not wish the size of the skew to be known. In this case, the bank might respond not with a specific cash price, but rather a discount quotation, an agreement to accept the equities in the basket at a particular volume-weighted average price, or other quotation based on the market prices of the equities after they are revealed. Because the bank can accurately assess its risk profile in accepting these, it can offer more competitive discounts or execution quotes for less risky baskets, or, similarly, charge more for a riskier basket.

The institution and the bank may agree to reveal:

- ☐ The full value of the long and short sides of the trade:

  The system provides a proof that allows the bank to decrypt the sum of all long trades and the sum of all short trades.

- The value or range of the "skew" only:

  In this case, the system provides the bank a proof of the sum of the portfolio's value plus all long positions' values minus all short positions' values. It might reveal the precise skew, or only that the skew lies within a particular interval.

- No information about the value of the incoming basket:

  In this case, the position values, quotes, and number of shares must all be kept secret; the risk profile of the resulting portfolio can still be evaluated by other means.

A similar approach can be applied to dividends, where the bank receives aggregate calculations of historical and expected dividend payments, so that it can estimate any dividend payments it will make (for short sales) and receive (for long positions).

9.3.6 Portfolio Composition Statistics

For risk management and hedging calculations, the bank may wish to know the composition of the combined portfolio based on various factors, including:

- Market sector (technology, health care, consumer goods, etc.)
- Market capitalization
- Index membership
- Dividend amount (as a percentage of share price)
- Average daily trading volume (possible in terms of both shares and notional value)

- Historical price volatility

Using our system, the institution need not reveal any information about the incoming basket's sector breakdown — for example, if there are balanced long and short trades in technology, and zero trades in utilities, this is indistinguishable to the bank from a portfolio with zero technology and balanced utilities trades, provided that the balanced trades do not change the risk profile of the bank's inventory. This provides additional secrecy to the institution while still meeting the needs of the bank.

The system calculates the portfolio composition and proves it to the accepting bank, who verifies the result using its own encrypted portfolio and the encrypted basket provided by the institution. Because the system can offer proofs that each sector's breakdown lies within a particular interval (say to the percentage point or 1/10 of 1%), the institution can reveal enough information for the bank to offer an accurate price while making reconstruction of the portfolio infeasible.

Using the general cryptographic operations described above (see Section 2.1.4), the bank now can compute verifiable breakdowns for the various aspects of the portfolio as follows.

We use the notation S to represent the total number of shares, and V to represent total portfolio value. Each of the ℓ elements' shares are written as $s_1, \ldots, s_\ell$; their total values are $v_1, \ldots, v_\ell$. In an example of a breakdown of market capitalization, s_1 might represent the sum of shares of securities whose market cap is over $10 billion, and s_10 represents microcaps of less than $50 million. Obviously, which "bucket" an equity belongs to is public information for any breakdown; the bank simply doesn't know the equities' quantities. The bank now has the encryptions of these values:

$E(S), E(V), E(s_i), E(v_i)$.

In the following example, the system proves the breakdown of equities based on the value of each bucket v_i in the breakdown. A similar technique can be applied to the number of shares, although the market value breakdown is generally more useful. We assume a constant K that reflects the desired granularity of the data as a number of "units"; for percentage points, the protocol would set $K = 100$.

Step 1. Because the bank knows the breakdown for each equity (e.g. market cap, market sector, etc.), it can compute encrypted sums of the number of shares and total value for each item in the breakdown by summing up the encrypted number of shares and total value from the combined portfolio. The bank also recalls the encrypted total number of shares and encrypted total value of the basket. We recall that this is the *combined* portfolio, where any long and short trades in the incoming basket have already been incorporated into the bank's inventory.

Step 2. The system first proves the sums are correct, namely, $E(S) \equiv \sum_{i=1}^{\ell}(s_i)$ and $E(V) \equiv \sum_{i=1}^{\ell} E(v_i)$. (In this case, $\sum$ represents applying the $\oplus$ operator to calculate an encryption of the sum of two encrypted values' plaintexts.)

Step 3. The system then prepares an encrypted "unit size" Z by computing Z such that[8] $ZK \leq V$ and $(Z+1)K > V$. The system proves this by providing the bank $E(Z)$ and a trivial encryption $E(K)$ and proving that $E(Z) \otimes E(K) \trianglelefteq E(V)$ and $(E(Z) \oplus E(1)) \otimes E(K) \triangleright E(V)$. Thus there are K "units" of size Z in the breakdown.[9]

[8] ZK was an unintended pun.

[9] Care must be taken so that $V \mod K$ is not too large, because this could skew the results. The system can even show the bank that value by revealing the verifiable result $E(V) \ominus (E(K) \otimes E(Z))$, or proving that it is less than a small constant. Since K is public, the bank can refuse a K that is too small.

Step 4. For each element of the breakdown, the system prepares an interval proof of how many "units" that element comprises. It begins by calculating and revealing two integer constants a_i, b_i and their "trivial" encryptions $E(a_i), E(b_i)$; the bank can verify these are correct encryptions. For example, a_i might be 10 and b_i 12, to show the result is between 10 and 12 units.

Step 5. The system completes the interval proof, showing that $E(a_i) \otimes E(Z) \trianglelefteq E(v_i) \trianglelefteq E(b_i) \otimes E(Z)$. This proves that $a_i Z \leq v_i \leq b_i Z$. This bounds the value of the portfolio in bucket i without revealing any further information.

Step 6. Steps 4 and 5 are repeated for each "bucket" in the breakdown until the entire portfolio has been classified. The bank might check that $\sum_i a_i \leq K \leq \sum_i b_i$ to be sure that the breakdown provided is appropriate.

9.3.7 Other Measurements of Risk

Because of the flexibility of the mathematical operations that can be performed on the recipient bank's basket and the incoming basket, other, more complicated risk measurements are possible. While the above examples are of completely linear functions, which permit the recipient to estimate the incoming baskets from the output risk characteristics and his own inputs, our system provides for computation of polynomial functions of modest degree by using repeated multiplications of encrypted values to calculate exponents. This permits the computation of more complex risk analysis measurements whose definition under our framework we leave for future work.

9.4 Pricing and Payment

Two types of prices must be computed: the price at which each security is valued when the exchange clears, and the price that the third parties charge for providing the market-clearing liquidity. We treat these in turn, referring to the winning third party (which might be a consortium) as the investment "bank". We note that if our second protocol is used independently between a single institution and one or more investment banks for proving characteristics about a single basket trade, the institution's basket functions as the remainder.[10]

Because each of the securities in the exchange is presumed to be traded on a primary market, we adopt the common practice in block trading to allow the primary market to dictate a fair market price for the securities at the time of trading. The financial industry uses many reasonable methods for price determination in block trading, and we do not advocate a particular pricing model over another—provided that the trading prices are determined in a manner exogenous to the exchange. Examples of these methods include average prices over time such as the volume-weighted average price (VWAP), or simply the midpoint of the bid and offer at the time the market clears.

After the proofs are obtained, the third parties have learned enough information to calculate a price for the incoming basket. They can accurately assess the changes in risk on their own inventories if they accept the basket, and by measuring those changes, estimate hedging costs for equities it will carry and execution costs for unwinding the trades it does not wish to keep.

[10]In fact, this is equivalent to our exchange with a single participant.

9.4.1 Compensating the Liquidity Providers

The bank can be compensated in many ways; the simplest is for the bank to quote a brokerage commission that it accepts for executing the trades. If the bank perceives greater risk, the bank can charge a higher commission. Other pricing mechanisms are possible: if the cash value of the portfolio is revealed, the bank can quote a price based on that; if the skew is not revealed, then the bank can quote a price based on a discount factor or volume-weighted price after the transaction is agreed on. The institution can choose among the various banks' offers, and notify the winner. Once the transaction is complete, the bank accepting the basket will be able to verify that the information provided was correct when it receives the remainder portfolio — but we reiterate that an advantage of our protocol is the banks that do not win still have convincing proof that the information was correct: the institution can't favor one bank over another.

Another interesting possibility is for the liquidity providers to publish deterministically verifiable valuation functions for their risk premium calculations. Using these, they can submit a representative portfolio to the exchange, obtain the changes in risk on their portfolio, then the exchange runs their calculations on the encrypted risk data and publishes a verifiable, encrypted result. These results would then be used to prove the payments correct, or could even be used in a verifiable sealed-bid auction to prove which of the liquidity providers' calculations yielded the most competitive bid for liquidating the remainder.

9.4.2 Simply Allocating Liquidation Costs

The commissions for obtaining this liquidity from the third parties must now be distributed among the institutions participating in the exchange. One approach, which reveals very little, is to distribute the costs among the participants according to their proportion of the notional value of trading across the exchange. Since revealing the proportion of an institution's share of the volume across the exchange also allows the institution to compute that volume, the exchange operator reveals the total notional value traded and proves that amount correct using the encrypted baskets. Each institution can then compute its proportion of the notional value and pay its share of the commission. However, this scheme benefits institutions who take advantage of more of the liquidity provided by the outside parties.

9.4.3 Fairly Allocating Liquidation Costs

Thus, while total cost sharing is simple and convenient, we also consider a slightly more involved "pay for what you use" model: each institution pays its share of the commission based only on the benefit it derived from the securities provided by the liquidity providers. In this method, institutions that use more of the remainder (instead of the other institutions) to fill their trades pay a greater share of the commission. At the extremes, an institution that trades securities which do not appear in the remainder pays nothing, while an institution who is the only one trading a particular security pays the entire share of the commission for that security.

We illustrate this method with an example which refers back to Table 9.2.1. For simplicity, we will assume that each security trades at a price of \$1, and the liquidity

provider charged a commission of $9000. The notional values of the four institutions' baskets are $1800, $1100, $1500, and $500, respectively; the remainder basket's value is $900. The exchange operator then publishes the encrypted amounts of commission paid based on the pro rata notional value traded of each security: $3000 for ABC, $0 for DEF, $2000 for GHI, $1000 for JKL, and $3000 for MNO. The operator proves that their sum is the (public) total commission.

Next, the exchange operator proves the total trading interest for each security by publishing encrypted sums of the absolute notional value of the orders in each basket: 700 for ABC, 1600 for DEF, 400 for GHI, 900 for JKL, and 1300 for MNO. Then, using the above methods, the exchange operator can publish an encrypted breakdown of the commission to be paid per share.[11] In this case, the commissions work out to $429 per 100 shares of ABC, $0 per 100 shares of DEF, $500 per 100 shares of GHI, $112 per 100 shares of JKL, and $231 per 100 shares of MNO; this yields a total overcharge of $14 due to rounding error.[12] The market operator proves that these encrypted prorated commissions are correct given the encrypted values already computed.

The market operator finally uses these encrypted prorated commissions to give each institution a verifiable share of its commission without revealing the magnitude of the securities traded by other traders or the composition of the remainder basket. For example, Institution 1 would pay

$$(5 \times 429) + (3 \times 0) + (0 \times 500) + (2 \times 112) + (8 \times 231) = 4217.$$

[11] Since the numbers do not divide evenly, the market operator can simply round up to the nearest integer and prove that the result is within a small error, that is, the difference between the total commission and the reported commission is small.

[12] If verifiable operations over encrypted rationals are employed, even this rounding error can be eliminated at a constant factor of additional computation cost.

The others would pay $1358, $3103, and $336, respectively, for their share of the costs in liquidating the remainder.

We sketch a final, possibly fairer method inspired by the Vickrey auction, but we reserve a full treatment and analysis for later work. In this model, an institution's share of the commission would be based on its impact on the market versus the marginal economy without its basket. Thus, institutions who *improved* the market by submitting a basket with opposite interest from the remaining baskets would pay very little (or perhaps even be paid!). Institutions who made the market more unbalanced by submitting a basket with interest in the same direction the remaining baskets would pay a greater share of the commission, because its trades would only be filled by means of the liquidity providers.

9.5 Keeping the Pool Safe

Although our methods are designed to provide transparency without revealing exploitable information, there remain ways in which unscrupulous traders might try to exploit the exchange we propose.

One misuse of our exchange might be for institutions to use its guaranteed liquidity to unload especially high-risk or illiquid securities. If the exchange becomes filled with undesirable assets, then banks will be less likely to want to participate. This is an important reason we advocate a pricing mechanism that charges institutions according to the amount of the remainder basket their trades represent—if the pricing mechanism is correctly defined, then institutions who submit less desirable portfolios will pay more for their liquidation costs.

Yet it might be desirable to make sure that the baskets the institutions submit to the exchange meet basic criteria for acceptability and portfolio risk. Using the same portfolio risk analysis techniques described above, institutions can issue zero-knowledge proofs about the baskets they submit so that all can be confident that their trades are acceptable. This should also reduce the third-party liquidation costs, because the third parties will be more secure that they aren't going to receive a basket that has nice overall characteristics but might be comprised of less desirable individual securities.

As we mentioned in the introduction, other common exploits associated with dark pools are less of a concern because our protocol features guaranteed execution. Exploits such as probing for existing liquidity and baiting (where someone places an order and then retracts it) are less of a problem, since once an order is placed, it cannot be retracted, and learning that your order was filled reveals nothing about existing opposite interest—every order is filled. Johnson [78] describes "toxic dark pools" that are known for being exploited.

9.6 Strengthening Secrecy

While the above solutions offer an appropriate degree of secrecy and are efficient to implement, the system does learn private data that it could reveal to others after the fact. It learns the trades that took place, which may be undesirable to certain institutions (notably hedge funds), and could learn something about the bank's inventory in the context of proving changes to the bank's risk without revealing the incoming portfolio characteristics directly. While the trades must eventually be reported to the

exchanges and become a matter of public record, and no such information could have any bearing on a particular round of the exchange, this information still has value. We thus consider how to mitigate the trust not to leak any information that we might place in the exchange operator.

The most compelling complement to our cryptographic solutions includes secure computing infrastructure such as Trusted Computing [142] hardware and network monitoring. We advance this idea in our previous work on cryptographic securities trading [150] and auctions proposals [115, 128]. In this scheme, specially designed hardware and software are trusted not to leak information, and monitored for security. We hasten to add that the secrecy-preserving correctness proofs we advance in this work complement such "black boxes" extremely well, because we need not trust the black box to produce correct results: we only use it to mitigate *ex post* disclosure. Thus, the actions of the exchange remain provably correct under all circumstances—even an undetected bug in the black box cannot result in incorrect behavior.

Even in these high-security settings, a determined adversary might be able to engineer steganographic leaks by "hiding" information in the protocol itself, often in predetermined bits of "random" help values. Doing so would be a significant effort, because most trusted computing infrastructures will not run software that has not been verified and signed by a third party, but we mention that small risk nonetheless. Fair Zero-Knowledge, introduced by Lepinski et al. [92], describes a mechanism to combat such attacks and surveys related work.

Finally, we observe that perfect security is never attainable in real life where humans are involved: any dishonest party "in the know" within any institution or bank

can always pick up the phone to deliver an out-of-band information leak. And, even where there is no intentional disclosure, Brandt and Sandholm proved impossibility results for achieving complete secrecy in some auction settings [34]. These ideas lead to interesting security questions about modern markets where more and more trades are performed without human input: automated trading agents running on secure hardware could offer an unprecedented level of security against the human element.

9.7 Conclusions and Future Work

We have implemented a useful new mechanism for block trading of securities that meets two market requirements: institutions can trade directly with each other when liquidity is available, while still having guaranteed execution for their entire order to limit portfolio and carrying risk. We employ a combinatorial exchange model, but make it tractable through external price discovery and a third party who provides necessary liquidity to achieve market equilibrium so that all orders are filled.

We protect the secrecy of sensitive data while giving the third party information necessary to calculate a fair commission by combining two novel cryptographic protocols. They are efficient, straightforward to understand, and can be implemented using already accepted cryptographic primitives.

More general formulations of these protocols may be of independent interest. Consider an arbitrary function over a finite field with encrypted inputs and a prover who proves facts about the output of this function. Clearly, there are many functions for which a precise output reduces the space of possible inputs dramatically — an unintended consequence of revealing a single output. Our mechanisms can offer prov-

ably correct yet *approximate* outputs using interval proofs, where exact results would reveal too much information.

The protocol we describe to prove changes to a recipient's risk also generalizes into a new class of price discovery. We can construct a more general protocol that allows a buyer to evaluate a purchase on the basis of a change in a buyer's utility function, rather than calculating the utility of the good directly. This means that in many business settings, where direct revelation of the good in question might have negative consequences, a buyer can engage in "zero-knowledge due diligence" where the buyer can satisfy many concerns by learning about how her utility function changes based on incorporating the good into her possessions, without learning enough about the good to allow the information to be exploited. These settings might include the sale of a significant commercial building, a business unit of a large corporation, or, other methods of trading financial instruments.

We leave for future work by ourselves and others a number of mechanism design questions. We believe it is possible to approach a true combinatorial exchange in which both institutions and liquidity providers post their desired baskets, where institutions post a maximum price they are willing to pay for liquidating their baskets, and whether and how their baskets are divisible; liquidity providers post "chunks" of liquidity associated with transaction costs for each chunk. The exchange then finds the optimal feasible allocation satisfying all possible atomic trades, and proves the outcome correct. Moreover, the use of such "chunks" could significantly reduce the size of any remainder basket, thereby reducing the size of any portfolio that needs to be traded blindly.

In addition to generalizing the protocols as described here, future work may also include a reference implementation of a prototype system or a more detailed technical specification based on a particular cryptosystem.

Chapter 10

Conclusions

There are excellent economic arguments for a compromise between perfect theoretical security and a completely trusted third party (Chapter 1). Our computational model, a partially trusted third party who is bound to always output correct results and employs tools from systems, not cryptography, to prevent leaking information after a computation's inputs are fixed, is a useful paradigm in which to design new cryptographic computational mechanisms, such as auctions or securities exchanges.

Because this dissertation is comprised of very different sections, we also include a "conclusions and related work" section in many of the chapters that refers to the specific contributions of that chapter. In this section, we consider the broad conclusions and our most general contributions to computer science and mechanism design.

We began with a general model of efficient, provably-correct, secrecy preserving computation in Chapter 2 and showed in Chapter 3 and a published paper [128] tools for implementing our general model. In Chapter 4 we outlined a new cryptographic primitive, time-lapse cryptography, that we first required to solve the problem of

nonrepudiation in auctions; we see many more uses for a service that provides such cryptographic keys.

We have constructed specifications for and prototypes of single- and multi-item sealed bid auctions under our model (Chapter 6) and shown them to be computationally practical while offering an acceptable degree of security. This gives us confidence that when we prototype our methods for combinatorial auctions (Chapter 7) and cryptographic single (Chapter 8) and combinatorial securities exchanges (Chapter 9), we will also find that our framework offers a valuable degree of security at a practical cost of computation and infrastructure.

We advance in Chapter 7 an extremely efficient mechanism for clearing cryptographic combinatorial auctions that is orders of magnitude faster than any we have seen in the literature. Our cryptographic securities exchanges work in Chapter 8 was among the first to consider in detail the complexities of the continuous double auctions in securities exchanges in a cryptographic setting. Our mechanism in Chapter 9 for operating a combinatorial securities exchange with guaranteed clearing through participating providers and price discovery through the primary markets is the first proposal we know of a mechanism that offers efficient, atomic clearing of basket orders—irrespective of the cryptography that may be employed to prove the actions of the exchange correct.

Throughout the work, we have established a strong link between cryptography and computational mechanism design, because cryptography offers unprecedented control over information. A market designer can now control *when* parties in a mechanism have information, *who* sees it, can limit disclosure to exactly what is needed for

the mechanism to be efficient—sometimes by revealing provably correct *approximate* views of private information. We propose the new term "cryptographic computational mechanism design"[1] as an appropriate name for the general class of research we advance in this work. We are not the first to see these relationships, though the field is very new; see Naor [109] for a survey of cryptography and mechanism design.

Many of our subprotocols are of independent interest outside the cryptographic model or particular mechanisms we describe. We have shown in Chapter 2 how to construct provably correct computations over the rationals using a set of primitive operations on the integers, an extension that applies to any secure integer computation model. We have also shown in Chapter 7 how to efficiently prove solutions to linear and integer programs correct without requiring the verifier to duplicate the entire computation.

Aside from the future work we consider in each chapter's own conclusions, we see significant opportunity for advancement of commercial and computational mechanisms through the use of the modern cryptographic tools we employ in this work. We have made but a small contribution with the commercial mechanisms we describe in this work, but we hope that the model and tools we have used will foster other, similar research to solve interesting open problems in computer science, finance, economics, law, and areas we have yet to imagine.

[1] While Xaotie Deng and Felix Brandt have also used the briefer term "cryptographic mechanism design", we recall the reader to the epigraphs of this dissertation's Preface.

Bibliography

[1] Dreyfus will pay $20.5 million to settle lawsuit. *The New York Times*, 22 June 2001.

[2] Settlement reached with five specialist firms for violating Federal securities laws and NYSE regulations. U.S. SEC Press Release, 2004. `http://www.sec.gov/news/press/2004-42.htm`.

[3] The mathematics genealogy project, May 2008. `http://www.genealogy.ams.org/id.php?id=8023`.

[4] Masayuki Abe. Mix-networks on permutation networks. In *Proc. ASIACRYPT '99*, volume 1716 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 1999.

[5] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 317–324. Springer, 2001.

[6] Masayuki Abe and Koutarou Suzuki. M+1-st price auction using homomorphic encryption. In *Proc. Public Key Cryptography*, 2002.

[7] Anat Admati and Paul Pfleiderer. Sunshine trading and financial market equilibrium. *Review of Finacial Studies*, 3:443–82, 1991.

[8] Jenny Anderson. S.E.C. is looking at stock trading. *The New York Times*, 6 February 2007.

[9] Edmund L. Andrews. Fed shrugged as subprime crisis spread. *The New York Times*, 18 December 2007.

[10] Jens Christopher Andvig. Corruption in the North Seal oil industry: Issues and assessments. *Crime, Law & Social Change*, 23:289–313, 1995.

[11] L. Arozamena and F. Weinschelbaum. The effect of corruption on bidding behavior in first-price auctions. Technical report, Universidad de San Andres, 2004.

[12] O. Ashenfelter. How auctions work for wine and art. *Journal of Economic Perspectives*, 3:23–36, 1989.

[13] John Asker and Estelle Cantillon. Properties of scoring auctions. Technical report, Leonard N. Stern School of Business, 2006.

[14] Association for the Advancement of Artificial Intelligence. *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, July 2000.

[15] Lawrence Ausubel, Peter Cramton, and Paul Milgrom. The clock-proxy auction: A practical combinatorial auction design. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 5. MIT Press, January 2006.

[16] Lawrence M. Ausubel and Paul Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1:1–42, 2002.

[17] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Token-controlled public key encryption. In *Proceedings of ISPEC 2005*, volume 3439 of *Lecture Notes in Computer Science*, pages 386–397. Springer Verlag, 2005.

[18] Vikas Bajaj. F.B.I. opens subprime inquiry. *The New York Times*, 30 January 2008.

[19] L. Baldwin, R. C. Marshall, and J.-F. Richard. Bidder collusion at forest service timber sales. *The Journal of Political Economy*, 105:657–699, 1997.

[20] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[21] O. Baudron and J. Stern. Non-interactive private auctions. In *Proc. Financial Cryptography*, 2001.

[22] Donald Beaver, Joan Feigenbaum, Rafail Ostrovsky, and Victor Shoup. Instance-hiding proof systems. Technical Report TR 93-65, DIMACS, 1993. DIMACS Center for Discrete Mathematics and Theoretical Computer Science.

[23] Mihir Bellare and Shafi Goldwasser. Verifiable partial key escrow. In *ACM Conference on Computer and Communications Security*, pages 78–91, 1997.

[24] Bruno Biais, Larry Glosten, and Chester Spatt. Market microstructure: a survey of microfoundations, empirical results and policy implications. *Journal of Financial Markets*, 8(2):217–264, May 2005.

[25] Ian F. Blake and Aldar C.-F. Chan. Scalable, server-passive, user-anonymous timed release cryptography. In *Proc. ICDCS '05*, pages 504–513, 2005.

[26] Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. A practical implementation of secure auctions based on multiparty integer computation. In *Proc. 10th International Conference on Financial Cryptography and Data Security (FC 2006)*, 2006.

[27] D. Boneh and M. K. Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology - CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 425–439. Springer Verlag, 1997.

[28] Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 68–77, New York, NY, USA, 2002. ACM.

[29] T. Börgers and E. van Damme. Auction theory for auction design. In M. C. W. Janssen, editor, *Auctioning Public Assets: Analysis and Alternatives*, chapter 1. Cambridge University Press, 2004.

[30] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Lecture Notes in Computer Science*, volume 1807, pages 431–444. Springer, 2000.

[31] Phillip G. Bradford, Sunju Park, and Michael H. Rothkopf. Protocol completion incentive problems in cryptographic Vickrey auctions. Technical Report RRR 3-2004, Rutgers Center for Operations Research (RUTCOR), 2004.

[32] Felix Brandt. How to obtain full privacy in auctions. Technical report, Carnegie Mellon University, 2005.

[33] Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, pages 201–216, 2006.

[34] Felix Brandt and Tuomas Sandholm. (Im)possibility of unconditionally privacy-preserving auctions. In *Proc. 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 810–817, 2004.

[35] Chad Bray. Two ex-Van der Moolen specialists are convicted of securities fraud. *The Wall Street Journal*, 15 July 2006.

[36] E. Brickell, D. Chaum, I. Damgård, and J. Van de Graaf. Gradual and verifiable release of a secret. In *Proceedings of CRYPTO'87*, volume LNCS 293, pages 156–166, 1988.

[37] Lianna Brinded. Nyse euronext steps into dark pool. *Dow Jones Financial News Online*, 24 Oct 2007.

[38] R. Burgeut and Y.-K. Che. Competitive procurement with corruption. *The RAND Journal of Economics*, pages 50–68, 2004.

[39] R. Burguet and M. Perry. Bribery and favoritism by auctioneers in sealed-bid auctions. Technical report, Institute of Economic Analysis, UAB, Barcelona, 2003.

[40] M. Burmester, E. Magkos, and V. Chrissikopoulos. Uncoercible e-bidding games. *Electronic Commerce Research*, 4:113–125, 2004.

[41] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proc. 6th ACM Conf. on Computer and Comm. Security*, pages 120–127, 1999.

[42] Ran Canetti and Marc Fischlin. Universally composable commitments. *Lecture Notes in Computer Science*, 2139:19, 2001.

[43] M. Celantani and J. J. Ganuza. Corruption and competition in procurement. *European Economic Review*, 46:1273–1303, 2002.

[44] A. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. In *Proceedings of EUROCRYPT'98*, volume LNCS 1403, pages 561–575, 1998.

[45] Xiaofeng Chen, Kwangjo Kim, and Byoungcheon Lee. Receipt-free electronic auction schemes using homomorphic encryption. In *ICISC*, 2003.

[46] Jung Hee Cheon, Nicholas Hopper, Yongdae Kim, and Ivan Osipkov. Timed-release and key-insulated public key encryption. Cryptology ePrint Archive, Report 2004/231, 2004.

[47] O. Compte, A. Lambert-Mogiliansky, and T. Verdier. Corruption and competition in procurement auctions. *The RAND Journal of Economics*, 36:1–15, 2005.

[48] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. *Lecture Notes in Computer Science*, 1592:311 ff., 1999.

[49] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Lecture Notes in Computer Science*, volume 1462, page 13 ff., 1998.

[50] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *Lecture Notes in Computer Science*, volume 1592, page 74 ff., 1999.

[51] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proceedings of Public Key Cryptography '01*, 2001.

[52] Robert W. Day and S. Raghavan. Fair payments for efficient allocations in public sector combinatorial auctions. *Management Science*, 2006.

[53] Giovanni Di Crescenzo. Privacy for the stock market. *Lecture Notes in Computer Science*, 2339:269 ff., 2002.

[54] N. Dimitri, G. Piga, and G. Spagnolo, editors. *Handbook of Procurement – Theory and Practice for Managers*. Cambridge University Press, 2006.

[55] Yevgeniy Dodis and Dae Hyun Yum. Time capsule signature. In *Proc. Financial Cryptography (FC '05)*, 2005.

[56] Kurt Eichenwald. Two sued by S.E.C. in bidding scandal at Salomon Bros. *The New York Times*, 3 December 1992.

[57] Kurt Eichenwald. Andersen guilty in effort to block inquiry on Enron. *The New York Times*, 16 June 2002.

[58] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, IT-31(4):469–472, 1985.

[59] W. J. Elmaghraby. Pricing and auctions in emarketplaces. In *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*. Kluwer Academic Publishers, Norwell, MA, 2004.

[60] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure multiparty computation of approximations. *Lecture Notes in Computer Science*, 2076:927 ff., 2001.

[61] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. *IEEE Symposium on Foundations of Computer Science*, pages 427–437, 1987.

[62] Michael J. Fishman and Kathleen M. Hagerty. The mandatory disclosure of trades and market liquidity. *The Review of Financial Studies*, 8(3):637 ff., 1995.

[63] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust efficient distributed RSA-key generation. In *Proceedings of the seventeenth annual ACM symposium on Principles of Distributed Computing*, page 320, 1998.

[64] Yair Frankel, Yiannis Tsiounis, and Moti Yung. "Indirect Discourse Proofs": Achieving efficient fair off-line E-cash. In Kwangjo Kim, editor, *Advances in Cryptology: Proceedings of ASIACRYPT 1996,* Kyongju, Korea, number 1163 in Lecture Notes in Computer Science, Berlin, 1996. Springer Verlag.

[65] M. K. Franklin and M. K. Reiter. The design and implementation of a secure auction server. *IEEE Transactions on Software Engineering*, 22(5):302–312, 1996.

[66] Gordon Gemmill. Transparency and liquidity: A study of block trades on the London Stock Exchange under different publication rules. *Journal of Finance*, 51:1765–1790, 1994.

[67] Rosario Gennaro. *Theory and Practice of Verifiable Secret Sharing.* PhD thesis, Massachusetts Institute of Technology, 1996.

[68] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Lecture Notes in Computer Science*, 1592:295 ff., 1999.

[69] Damien Giry and Philippe Bulens. Cryptographic key length recommendation. `http://www.keylength.com`, 2006.

[70] Shafi Goldwasser, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity, and a methodology of cryptographic protocol design. In *Proc. Symposium on Foundations of Computer Science (FOCS)*, pages 39–48, 1986.

[71] Daniel A. Graham and Robert C. Marshall. Collusive bidder behavior at single-object second price and english auctions. *Journal of Political Economy*, 95:1217–1239, 1987.

[72] M. Harkavy, J. D. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *Proc. Third USENIX Workshop on Electronic Commerce*, pages 61–74, 1998.

[73] Larry Harris. *Trading and Exchanges.* Oxford University Press, 2003.

[74] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *ACM Conference on Computer and Communications Security*, pages 100–110, 1997.

[75] A. T. Ingraham. A test for collusion between a bidder and an auctioneer in sealed-bid auctions. *Contributions to Economic Analysis and Policy*, 4:1–32, 2005.

[76] M. C. W. Janssen, editor. *Auctioning Public Assets: Analysis and Alternatives*. Cambridge University Press, 2004.

[77] Jeromee Johnson. Personal communication, 25 January 2008.

[78] Jeromee Johnson and Larry Tabb. Groping in the dark: Navigating crossing networks and other dark pools of liquidity, 31 January 2007.

[79] Mads J. Jurik. *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*. PhD thesis, University of Århus, 2003.

[80] Donald B. Keim and Ananth Madhavan. The upstairs market for large-block transactions: Analysis and measurement of price effects. *Review of Finacial Studies*, 9:1–36, 1996.

[81] Aggelos Kiayias and Moti Yung. Efficient cryptographic protocols realizing e-markets with price discrimination. In *Financial Cryptography and Data Security*, pages 311–325, 2006.

[82] Hiroaki Kikuchi. (m+1)st price auction protocol. In *Proc. Financial Cryptography*, 2001.

[83] Sevket Alper Koc and William S. Neilson. Bribing the auctioneer in first-price sealed-bid auctions. Technical report, Kocaeli University, 2006.

[84] Anshul Kothari, David C. Parkes, and Subhash Suri. Approximately-strategyproof and tractable multi-unit auctions. *Decision Support Systems*, 39:105–121, 2005.

[85] Vijay Krishna. *Auction Theory*. Academic Press, 2002.

[86] Manoj Kumar and Stuart I. Feldman. Internet auctions. In *Proc. 3rd USENIX Workshop on Electronic Commerce*, 1998.

[87] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proc. 38th Annual Symposium on Foundations of Computer Science (FOCS 1997)*, pages 364–373, 1997.

[88] J.-J. Laffont and J. Tirole. Auction design and favoritism. *International Journal of Industrial Organization*, 9:9–42, 1991.

[89] S. Lahaie. An analysis of alternative slot auction designs for sponsored search. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 2006.

[90] Y. Lengwiler and E. Wolfstetter. Bid rigging. An analysis of corruption in auctions. Technical report, Humboldt-University at Berlin, 2005.

[91] Y. Lengwiler and E. Wolfstetter. Corruption in procurement auctions. In N. Dimitri, G. Piga, and G. Spagnolo, editors, *Handbook of Procurement – Theory and Practice for Managers*, chapter 16. Cambridge University Press, 2006.

[92] Matt Lepinski, Silvio Micali, and abhi shelat. Fair zero-knowledge. In *Proc. Theory of Cryptography Conference*, pages 245–263, 2005.

[93] LiDIA-Group. LiDIA — a library for computational number theory. *TU Darmstadt*, 2001.

[94] H. Lipmaa, N. Asokan, and V. Niemi. Secure Vickrey auctions without threshold trust. In *Proc. 6th International Conference on Financial Cryptography (FC 2002)*, pages 87–101, 2002.

[95] Ananth Madhavan. Market microstructure: A survey. 8, March 2000.

[96] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - a secure two-party computation system. In *Proc. of USENIX Security Symposium*, pages 287–302, 2004.

[97] Silvio Micali Manuel Blum, Paul Feldman. Non-interactive zero-knowledge and its applications. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 103–112. ACM Press, 1988.

[98] Shin'ichiro Matsuo and Hikaru Morita. Secure protocol to construct electronic trading. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, E84-A(1):281–288, 2001.

[99] Timothy C. May. Timed-release crypto. In *The Cyphernomicon: Cypherpunks FAQ and More, v. 0.666*, chapter 14.5. September 1994.

[100] P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[101] J. McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8:145–162, 1994.

[102] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[103] F. M. Menezes and P. K. Monteiro. Corruption and auctions. Technical report, Getulio Vargas Foundation, Rio de Janeiro, Brazil, 2001.

[104] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, April 1978.

[105] Paul Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[106] B. Moldovanu and M. Tietzel. Goethe's second-price auction. *Journal of Political Economy*, 106:854–859, 1998.

[107] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

[108] Toru Nakanishi, Daisuke Yamamoto, and Yuji Sugiyama. Sealed-bid auctions with efficient bids, 2003.

[109] Moni Naor. Cryptography and mechanism design. In *Proc. 8th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 163–167, 2001.

[110] Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *Proc. First ACM Conf. on Elec. Commerce*, pages 129–139, 1999.

[111] Minh-Huyen Nguyen, Shien Jin Ong, and Salil Vadhan. Statistical zero-knowledge arguments for np from any one-way function. In *Proceedings of Foundations of Computer Science*, pages 3–14, 2006.

[112] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. Cambridge University Press, 2006.

[113] Pascal Paillier. *Cryptographie à Clé Publique Basée sur la Résiduosité de Degré Composite*. PhD thesis, École Nationale Supérieure des Télécommunications, 1999.

[114] Pascal Paillier. Public-key cryptosystems based on composite residuosity classes. In *Proc. EUROCRYPT '99*, pages 223–239, 1999.

[115] D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. A. Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 70–81, New York, NY, USA, 2006. ACM Press.

[116] David C. Parkes, Ruggiero Cavallo, Nick Elprin, Adam Juda, Sebastien Lahaie, Benjamin Lubin, Loizos Michael, Jeffrey Shneidman, and Hassan Sultan. ICE: An iterative combinatorial exchange. In *ACM Conf. on Electronic Commerce*, pages 249–258, 2005.

[117] David C. Parkes, Jayant R. Kalagnanam, and Marta Eso. Achieving budget-balance with Vickrey-based payment schemes in combinatorial exchanges. Technical report, IBM Research Report RC 22218, 2001.

[118] David C. Parkes, Jayant R. Kalagnanam, and Marta Eso. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proc. Fourth ACM Conf. on Electronic Commerce*, pages 1161–1168, 2001.

[119] David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher A. Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. *Electronic Commerce Research and Applications*, 2008. To appear.

[120] David C. Parkes and Lyle H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)* [14], pages 74–81.

[121] David C. Parkes and Lyle H. Ungar. Preventing strategic manipulation in iterative auctions: Proxy agents and price-adjustment. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)* [14], pages 82–89.

[122] T. P. Pedersen. Non-interactive and information theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO*, Lecture Notes in Computer Science, pages 129–140. Springer Verlag, 1991.

[123] M. Pesendorfer. A study of colllusion in first-price auctions. *The Review of Economic Studies*, 67:381–411, 2000.

[124] David Porter, Stephen Rassenti, Anil Roopnarine, and Vernon Smith. Combinatorial auction design. *Proceedings of the National Academy of Sciences*, 100(19):11153–11157, 2003.

[125] Robert H. Porter and Douglas J. Zona. Detection of bid-rigging in procurement auctions. *Journal of Political Economy*, pages 518–538, 1993.

[126] M. O. Rabin. Digitalized signatures. In *Foundations of Secure Computing*, pages 155–166. Academic Press, New York, 1978.

[127] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[128] Michael O. Rabin, Rocco A. Servedio, and Christopher Thorpe. Highly efficient secrecy-preserving proofs of correctness of computations and applications. In *Proc. IEEE Symposium on Logic in Computer Science*, 2007.

[129] Michael O. Rabin and Christopher Thorpe. Time-lapse cryptography. Technical Report TR-22-06, Harvard University School of Engineering and Computer Science, 2006.

[130] Barbara Rindi. Transparency, liquidity and price formation. In *Proceedings of the 57th European Meeting of the Econometric Society*, 2002.

[131] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Technical Report MIT/LCS/TM-82, MIT Laboratory for Computer Science, 1977.

[132] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT, 1996.

[133] Marc S. Robinson. Collusion and the choice of auction. *Rand J. Econ.*, 16:141–145, 1985.

[134] Susan Rose-Ackerman. The economics of corruption. *Journal of Public Economics*, 4:187–203, 1975.

[135] Michael H. Rothkopf, Thomas J. Teisberg, and Edward P. Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98:94–109, 1990.

[136] T. C. Salmon. Preventing collusion between firms in auctions. In M. C. W. Janssen, editor, *Auctioning Public Assets: Analysis and Alternatives*, chapter 3. Cambridge University Press, 2004.

[137] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2002.

[138] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, 2005.

[139] T. C. Schelling. *The Strategy of Conflict.* Harvard University Press, 1980.

[140] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[141] Shepherd Smith & Edwards. Citigroup, Merrill Lynch and Lehman exbrokers face retrial in eavesdropping case. *Stockbroker Fraud Blog*, 27 May 2007. `http://www.stockbrokerfraudblog.com/2007/05/exbrokers_of_citigroup_merrill_1.html`.

[142] Sean W. Smith. *Trusted Computing Platforms: Design and Applications*. Springer, New York, 2005.

[143] T. Smith, T. Sandholm, and R. Simmons. Constructing and clearing combinatorial exchanges using preference elicitation. In *AAAI-02 workshop on Preferences in AI and CP: Symbolic Approaches*, 2002.

[144] Hans R. Stoll. Market microstructure. In G. M. Constantinides, M. Harris, and R. Stulz, editors, *Handbook of the Economics of Finance*. Elsevier Science B.V., 2003.

[145] S. G. Stubblebine and P. F. Syverson. Fair on-line auctions without special trusted parties. In *Proc. of Financial Cryptography*, 1999.

[146] Koutarou Suzuki and Makoto Yokoo. Secure combinatorial auctions by dynamic programming with polynomial secret sharing. In *Proc. 6th Int. Financial Crypto. Conf.*, 2002.

[147] Koutarou Suzuki and Makoto Yokoo. Secure generalized Vickrey auction using homomorphic encryption. In *Proc. Financial Cryptography*, 2003.

[148] Michael Szydlo. Risk assurance for hedge funds using zero knowledge proofs. In *Proc. 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, 2005.

[149] The World Bank. *Guidelines Procurement Under IBRD Loans and IDA Credits*. The International Bank for Reconstruction and Development, The World Bank, Washington, D.C., 2006.

[150] Christopher Thorpe and David C. Parkes. Cryptographic securities exchanges. In *Proc. Financial Cryptography and Data Security*, 2007.

[151] Peter Trepte. *Regulating Procurement*. Oxford University Press, 2004.

[152] U.S. Central Intelligence Agency. The World Factbook: European Union. `https://www.cia.gov/library/publications/the-world-factbook/print/ee.html`, 2007.

[153] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[154] C. Wang, H. Leung, and Y. Wang. Secure double auction protocols with full privacy protection. In *Information Security and Cryptography - ICISC 2003: 6th International Conference*, 2003.

[155] Laurence A. Wolsey. *Integer Programming*. John Wiley, 1998.

[156] M. Yokoo and K. Suzuki. Secure generalized vickrey auction without third-party servers. In *Proc. Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2004.

[157] Makoto Yokoo and Koutarou Suzuki. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In *Proc. First Int. Conf. on Autonomous Agents and Multiagent Systems*, 2002.

[158] Makoto Yokoo and Koutarou Suzuki. Secure generalized vickrey auction without third-party servers. In *Eighth International Financial Cryptography Conference (FC-2004)*, pages 132–146, 2004.