P4P: A Practical Framework for Privacy-Preserving Distributed Computation

Yitao Duan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-165
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-165.html

December 19, 2007

Acknowledgement

I would like to thank my advisor, Professor John Canny, for introducing me to the area of privacy technologies and guiding me through my research. I would like to thank Professor Doug Tygar and Professor John Chuang for serving on my qualifying exam and dissertation committee. I also would like to thank Professor David Wagner for serving on my qualifying exam committee. All these faculties provided valuable feedbacks on my research proposal and helped me formulate my research plan.

P4P: A Practical Framework for Privacy-Preserving Distributed Computation

by

Yitao Duan

B.S. (Beijing University of Aeronautics and Astronautics) 1994
M.S. (Beijing University of Aeronautics and Astronautics) 1997

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy
in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2007

The dissertation of Yitao Duan is approved.

Chair	Date
	Date
	Date

University of California, Berkeley

Fall 2007

P4P: A Practical Framework for Privacy-Preserving Distributed Computation

Abstract

P4P: A Practical Framework for Privacy-Preserving Distributed Computation

by

Yitao Duan

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor John Canny, Chair

Privacy is becoming an increasingly important issue in electronic commerce and other online activities that are growing in popularity. This work introduces a framework, called Peers for Privacy (P4P), for implementing many useful algorithms with provable privacy and adequate efficiency in a realistic adversary model at a reasonably large scale. The basic idea is to decompose an algorithm into a series of addition-only steps, which have very efficient private implementation using cryptographic tools. This simple model is surprisingly general and supports many algorithms prevalent in distributed data mining. Examples include linear algorithms like voting and summation, as well as nonlinear algorithms such as regression, classification, SVD, PCA, k-means, ID3, machine learning algorithms based on Expectation Maximization (EM), etc. In fact all algorithms in the statistical query model are supported.

The computation of the sums is based on a highly efficient verifiable secret sharing (VSS) scheme that allows secret-shared arithmetic operations to be done over *small* fields (e.g. 32 or 64 bits) where private arithmetic operations have the same cost as normal arithmetic. This thesis shows that this paradigm admits efficient zero-knowledge tools that can be used to verify the properties of user data such as equality and boundedness. These tools provide practical mechanisms to deal with cheating users. One such tool is an extremely efficient zero-knowledge proof that verifies the L2-norm of the user data is bounded by a constant. This is to prevent a malicious user from exerting too much influence on the computation. The verification uses a linear number of inexpensive small field operations, and only a logarithmic number of large-field (1024 bits or more) cryptographic operations, and can achieve orders of magnitude reduction in running time over standard techniques (from hours to seconds) for large-scale problems.

Concrete examples are given to demonstrate how the framework supports private computation of popular algorithms such as SVD, link analysis and association rule mining. The thesis also includes schemes for scalable multicast encryption and bidirectional group communication. They provide secure data transmission support for the type of communication pattern required by the P4P framework and many other group-oriented applications.

Professor John Canny
Dissertation Committee Chair

To my parents, my wife, Dan Huang, and my son, Shaoxiong the Little Bear.

Contents

List of Figures

List of Tables

Acknowledgements

It would be impossible for me to finish this work and obtain my Ph.D. without the help and support from many people. I would like to take this opportunity to express my gratitude towards them.

First and foremost, I would like to thank my advisor, Professor John Canny, for introducing me to the area of privacy technologies and guiding me through my research. His insights and vision have been of immense value to me. I learned from him the importance of applying one's research to solving real-world problems and paying attention to practical issues. His trust and unrestricted support allow me to explore the research problems that intrigue me most. His advices and insightful questions shed light on the problems and help me reach solutions. This work will not materialize without his valuable guidance.

I would also like to thank Professor Doug Tygar and Professor John Chuang for serving on my qualifying exam and dissertation committee. Their early feedback to my thesis proposal was very helpful for me to formulate my research plan. I am also very grateful to them for reading and commenting on my thesis.

I would like to thank Professor David Wagner for serving on my qualifying exam committee. I learned a lot from his two excellent courses: computer security and cryptography. They provide me with background knowledge necessary for my thesis work. I also benefit tremendously from discussions with him on my research in general and the multicast encryption scheme in particular, which is included in chapter 6 of this thesis.

I would like to thank my colleagues in the Berkeley Institute of Design (BiD) for their insightful comments and intriguing discussion: Jingtao Wang, Matthew Kam, David Nguyen, Ana Ramirez Chang, Tye Rattenbury, Jeremy Risner, Jonathan Hey, Omar Khan, Andy Carle and David Sun.

My friends in Berkeley make these years full of joy and excitement. Ling Huang, Jialing Lang, Wei Liu and Nan Ding, we shared many joyful and memorable trips together. Bo Zhu gave me tremendous help in my early days in Berkeley. I am grateful to my friends in EECS and other departments for the happy time we shared together: Yunfeng Li, Feng

Zhou, Li Zhuang, Wei Xu, Jimmy Su, Hao Zhang, Qi Zhu, Weidong Cui, Kai Wei, Li Yin, Fang Yu, and many others.

Finally, I am extremely grateful to my parents and my wife. Without their selfless love and support, I wouldn't be able to reach this point in my life. They supported me with sacrifice and love during the most difficult time of my life when I was struggling with physical illness. My wife, Dan Huang, accompanied me during all the years in Berkeley and endured all the difficulties with great courage. I will be eternally grateful for her love, care, support, and encouragement.

Chapter 1

Introduction

Many kinds of analysis depend on mining data from a group of users. Examples include linear algorithms like voting and summation, as well as non-linear ones such as regression, SVD, k-means, ID3, and many machine learning algorithms based on the EM (Expectation Maximization). These analysis methods can be found in a wide range of applications from E-commerce to medical research. We consider a fully distributed setting where each user holds her own data and participates in mining tasks managed by a few data miners. One example scenario could be e.g. two research institutes analyzing user survey data. In all these cases, it is very important to protect user privacy to the maximum extent possible. Indeed in some situations laws even prohibit service (e.g. health care) providers from disclosing customers information, even to other providers. In this case even if users are willing to cooperate, service providers are restricted to mining data from their own customers, which may not allow them to discover valuable results that would have been available by mining across-providers data. At the same time, it is important to perform the same computation with no loss of accuracy (which can mean lost revenue for the provider, or misleading results for research institutes). Providers do not benefit directly from privacy technology, so the costs to them must be as small as possible, and ideally zero. These are the constraints that have guided the work reported here.

Privacy-preserving data mining has been an active area of research since it was introduced by Agrawal and Srikant [4] and Lindell and Pinkas [83]. Existing solutions use either randomization (e.g. [50, 46]) or cryptographic techniques (e.g. [83, 45, 123, 132, 130]) to protect privacy. Besides sacrificing accuracy, randomization has been shown to provide very little privacy protection in many cases unless sufficient amount of noise is introduced [72, 11, 43]. Works in the second category typically make use of private computation protocols and enjoy provable privacy provided by the cryptographic primitives. However, there has been two major inadequacies with almost all existing cryptography-based solutions: (1) the schemes are not practical for large scale systems due to their heavy use of expensive cryptographic operations, and (2) there is generally no efficient mechanism to handle actively cheating users. All the works in [83, 45, 123, 132, 130] deal with only *passive* adversary who never deviates from specified behavior. The assumption is certainly not realistic when the data comes from individual users, some of whom may incentivized to bias the computation or have their machines corrupted by hackers. Securing the system against such adversary is a necessary step in incorporating privacy-preserving data mining into real-world applications.

To summarize, for a privacy solution to be practical, it must satisfy the following, which existing schemes fail to achieve except for very rare applications:

Privacy It must provide provable privacy.

Efficiency and Scalability It must have adequate efficiency at reasonably large scale.

Robustness It must be secure against realistic adversaries. Since the data is from the users, it must be able to handle actively cheating users.

Cryptography provides primitives with various level of efficiency [33, 58, 34, 21]. While both addition and multiplication are possible in most of these schemes, the practical overhead for multiplication is much higher. Thus one way to build a practical private computation scheme is to avoid the multiplication and focus on algorithms that can be implemented using addition-only steps. Surprisingly, a lot can be done with such an approach. The standard algorithms for many of the above-mentioned analysis use gradient steps which sum vector data from the users. These steps are *linear* in per user data and can be implemented using an addition-only approach. Implementing the algorithms using summation forms has been used by other works as a general approach to parallelize the algorithms and run them in a distributed fashion. For example, [29] and [37] showed that many popular algorithms has a summation implementation that can be computed with Google's MapReduce framework, which is a distributed programming construct over clusters that is being successfully deployed in production. The examples included an EM algorithm for pLSI [69], Locally Weighted Linear Regression (LWLR), Naive Bayes (NB), PCA, etc. In fact all the algorithms in the statistical query model [75] can be expressed in this form. They demonstrated the versatility of vector-addition in implementing statistical learning algorithms. This phenomena has also been observed by researchers in privacy technology field and used as a way to implement the algorithms with privacy. These works include private SVD [21], EM-based collaborative filtering [22], link analysis algorithms such as HITS algorithm [48].

Existing private addition-only solutions are still not always practical due to their heavy use of public-key operations for information hiding and/or verification. While these examples have the same asymptotic complexity as the standard algorithms for those problems, the constant factors imposed by public-key operations are prohibitive for large scale systems. On a typical computer today there is a six order of magnitude difference between the crypto exponentiations (in large field) needed for secure homomorphic computation (order of milliseconds) and regular arithmetic operations in small (32- or 64-bit) fields (fraction of a nano-second). Both homomorphic arithmetic [34, 21] and VSS (Verifiable Secret Sharing) [28] rely on public-key operations for verification. Even fast-track VSS [58] does not reduce the asymptotic number of crypto operations.

1.1 Our Approach

We propose a new approach to address the above-mentioned difficulties and provide practical privacy solutions for real-world applications. Our approach is called Peers for Privacy (P4P) for its unique architecture and privacy mechanism. The main computation is based on secret sharing over *small* field, which avoids the manipulation of large integers so that private computation on each participant has the *same* cost as regular non-private computation. To address the issue of cheating users, we provide a number of very efficient probabilistic zero-knowledge protocols that can be used to verify the properties of user data such as equality and boundedness. One such tool is an extremely efficient zero-knowledge proof for verifying that the L2-norm of the user data is bounded by a constant. This is to prevent a malicious user from exerting too much influence on the computation. The verification uses a linear number of inexpensive small field operations, and only a *logarithmic* number of large-field (1024 bits or more) cryptographic operations (in the size of the user data). This improves significantly over standard techniques that require at least linear number of such expensive steps. Our scheme can achieve orders of magnitude reduction in running time (from hours to seconds) for large-scale problems. In our scheme, the cost will be dominated by the linear number of small-field operations that one has to pay even when the computation is done directly on user data without privacy. In a sense, privacy is almost "free" even counting constant factors.

Our basic approach, VSS-based private computation over small field and random projection-based method for verifying (in zero-knowledge) high-dimensional data, also opens doors to many other practical ZK tools. These include a zero-knowledge vector equality test and potentially many others. These protocols also use only a small (constant or logarithm in the size of input) number of large field operations so that they preserve the efficiency we obtained by using small field VSS. Such tools provide practical means to deal with realistic adversary models which accommodate active cheating from the users.

1.2 Peers for Privacy: Overview

In P4P, we assume there is a single computer called the *server*, which is operated by a service provider or a data miner. Unlike other server-based solutions such as [132], P4P also involves a (small) number of designated parties called *privacy peers* (PP) to participate in the computation. In contrast to previous work, privacy peers do not have to be dedicated servers and can belong to users in the community. The fundamental role of the privacy peers is to, by participating in the computation, offload the some information, thus trust, from the server such that no one in the system, except for the owner herself, has access to user's data. Privacy peers are not required to be honest, and the protocol ensures that they cannot break the privacy of the protocol without the server's help.

This architecture is a hybrid of client-server and P2P. On one hand, the server shares the bulk of the computation and storage, and also synchronizes the protocol. This allows us to take advantage of its large computation/storage capacity and high availability. It also leads to practical, efficient protocols that are not possible with fully distributed architecture. On the other hand, the peers also participate in the computation and provide privacy. This particular architecture is motivated by several practical considerations, not all of them technological (privacy has never been a purely technological issue anyway):

- Client-server is the de facto business and service model for most existing service providers. It is unlikely that they will be willing to abandon their existing infrastructure and practice simply for the sake of providing users with privacy.
- In a purely serve-based system, there is a tremendous power imbalance between the service provider, who possesses all the information and controls who has access to it, and the client users. This imbalance is prevalent in almost all aspects of their relationship including economic, information access, and computing etc.
- Any privacy enhancing technology will inevitably reduce the amount of user information a service provider can collect and entail additional cost. Understandably, the service providers will be reluctant to adopt them. For any technological solutions to be viable, (1) it must be efficient: the cost for privacy protection should be small, ideally free, compared to the main computation; (2) the service provider's incentives must be taken into consideration.
- The power imbalance and the lack of incentive can be offset by the active participation of users/peers. At minimum, privacy protection requires user awareness and control of the collecting and use of personal information. Given the dominance and the

reluctance on the server side, users must be involved in the loop to secure and verify the protection.

- Current distributed multiparty computation protocols have computation and communication complexity at least quadratic in the number of participants, with large hidden constant in the asymptotic complexity, and are not practical for large scale systems. But for many applications, the more users involved, the better quality the computation results are. Such applications are prevalent in data mining. The minimal scale required by such applications makes it infeasible to carry out the computations among the users.

P4P accommodates service provider's existing business model while taking advantage of participants' heterogeneity that exists in real-world systems. The server, which is powerful and highly available, is still responsible for most of the computation/storage but the privacy peers work with the users to provide privacy. This arrangement allows us to use a VSS-based paradigm for computation which can be executed in regular-sized (32- or 64-bit) fields. Big integer field is used only for verification which involves only a small number (constant or $O(\log m)$ where m is the size of user data) of big integer operations. This avoids the pitfall of generic MPC protocols (e.g. [10, 60, 6, 64, 58]) which work in the big field all the time. In P4P the cost is dominated by the *regular* arithmetic that one would have to pay even in a server-based, non-private scheme. In addition, P4P computes a function over data from all n users but it does not use an n-party protocol. Instead, computation in P4P consists of a number of independent, parallel pairwise two-party protocols between the server and one privacy peer. Each privacy peer services a small subset of users and each pairwise computation produces aggregate over this subset and the server further aggregate them to produce the final sum. Compared to a fully distributed multiparty scheme, by restricting the number of players to 2 in each pairwise computation, many expensive operations such agreement are made efficient. We can show that private computation in P4P is almost as efficient as regular, centralized arithmetic operations. In this sense P4P add privacy at almost no cost to the service provider and provides a practical solution for large scaled private computation tasks.

The candidates for the privacy peers may differ with applications. Since the they are not trusted and cannot compromise user data, there are many options. It has been demonstrated that in practical P2P systems such as Gnutella and Napster, a small fraction of the users in the community provide most of the services to the others. The existence of such altruistic users is a pervasive phenomenon in communities and those users can provide the computation cycles for the privacy peers. In this case, we assume a much smaller ratio between peer and users, e.g. while a single server would normally service millions of users, an individual privacy peer would support a few hundreds to perhaps tens of thousands of users. For workplace privacy, the peer would be a special employee (e.g. a union representative). The service may also be provided for a fee by a third-party commercial "privacy provider". In certain cases, it may be feasible to distribute the VSS among service providers. For example, two hospitals may wish to mine data collected from patients. However, it is not desirable to disclose patients' private information. In this case P4P can be used to support this type of data mining by letting one of the service providers assume the role of a privacy peer in the protocol and the result is that both hospitals learn the accurate (aggregate) final computation but neither learns anything about users private data.

P4P does not overthrow existing systems. Instead, it offers a way to "patch" them, with minimum impact on performance, to provide provable privacy for the users while maintaining the accuracy of the computation. This approach is applicable to a large number of existing and emerging real-world applications and is more realistic to guarantee adoption.

1.2.1 Security from Heterogeneity

The security of a P4P system is based upon the assumption that the server and the privacy peers won't collude (the corrupted privacy peers may collude with cheating users or other peers). We show why this is realistic.

Collusion comes in two forms:

1. The owners of the two machines conspire and share data and coordinate their actions;

2. An attacker (who can be one of the two players) corrupts both machines.

These two attacks, although often handled uniformly in most schemes, rely on different conditions to be successful: for two players to conspire, they must have the right incentive, which may come from anticipated benefit out-weighting foreseeable risk, and overcome the mutual distrust between them. For an adversary to corrupt both machines, it must penetrate the armor of the better protected one. P4P defends against them with different mechanisms, targeting at their individual conditions. Concretely, we make the following observations/assumptions:

1. The server is well protected against outside attacks.

2. The server and the privacy peers won't conspire and share data between each other.

3. The owner of the server won't attempt to break into peers' machines.

All three are provided by exploiting the asymmetry between server and privacy peer, which is an important source of security. While peer machines are not always well maintained, cooperations invest large amount of resources protecting their servers. In terms of incentives, a service provider will benefit from accessing sensitive data from many users, but much less so from data from a few users. To gain any substantial advantage, the server must collude with or break into a large number of peers. This can hardly be done without being detected: corrupted peers will exchange data with the server, and all will be aware of the cheating. Any of these peers may, deliberately or accidentally, expose the server's cheating or discover the break-in.

In essence, by exploiting the difference in their incentives, the heterogeneity of their individual protection, and the mutual *distrust* among the participants, P4P is made efficient yet secure. In a nutshell, our system relies on the server for defending against external attacks and uses the privacy peers to protect user privacy against a curious server. In practice the protection offered by this arrangement is very good compared to multi-server approaches. And economically, the P4P approach is superior since it requires no additional resources on the server side.

1.3 The P4P Framework

We describe our protocols as 2-way multi-party computations carried out between the server and a privacy peer who are called *talliers*. In this work we denote the server T_1 and the privacy peer T_2. The entire P4P computation is composed of a number of such independent pair-wise computations.

We define a *server* as all the computation units under the control of a single entity. It can be a cluster of thousands of machines so that it has the capability to support a large number of users. From information-sharing perspective, it suffice to view all the machines under a single control as a single entity which is referred to as a server in this work.

This is a slightly generalized version of the models many privacy-preserving data mining schemes consider (e.g. [83, 45, 123, 132, 130])). In their models, each server holds either a subset of attributes of all users, or all attributes of a subset of users. The former is called vertical partition [122, 123] and the latter horizontal partition [71, 83]. In our model, however, no user data ever leaves its owner unprotected and no server can obtain any information about user data other than what can be inferred from the final results as long as at least one server is uncorrupted. Arguably, this offers much better privacy protection since it conforms to the "user-owned and operated" privacy principle that is both natural and effective in many situations [21]. All the other models can be easily transformed into ours by letting the servers act on users data on behalf of the owners. Formally the threat model is defined as follows:

Threat Model *Let α be the upper bound on the fraction of the dishonest users in the system.* [1] *We consider a computationally bounded adversary whose capability of corrupting parties is modelled as follows:*

1. *An adversary may* actively *corrupt at most $\lfloor \alpha n \rfloor$ users, taking full control of their machines and causing them to deviate arbitrarily from the specified protocol.*

[1] *Most mining algorithms need to bound the amount of noise in the data to produce meaningful results. This means that the fraction of cheating users must be below a reasonable threshold (e.g. $\alpha < 20\%$). Under this condition our scheme should give the honest players fairly good privacy protection.*

2. *In addition to 1, we also allow the* same *adversary to passively corrupt one of the talliers. When a party is passively corrupted, all her data is exposed to the adversary but the party continues to follow the protocol.*

This threat model is similar to that of [53] in that some of the participants are actively corrupted while some others are passively corrupted by the same adversary *at the same time*. This is an extension to the general adversary structure introduced by Hirt and Maurer [67, 68]. Our model does not satisfy the feasibility requirements of [67, 68] and [53]. We avoid the impossibility by considering addition only computation.

The model models realistic threats in our target applications. In general, users are not trustworthy. Some may be incentivized to bias the computation (e.g. to drive down/up the price of an item), some may have their machines corrupted by hackers. So we model them as active adversaries and our protocol ensures that the active cheating from a small number of users will not exert large influence on the computation. This greatly improved over existing privacy-preserving data mining solutions (e.g. [83, 132, 130, 123]) which handle only purely passive adversary. The corrupted tallier, on the other hand, is modelled as a passive adversary that can share data with corrupted users. This captures our incentive and non-collusion assumption mentioned earlier.

REMARKS In this thesis, we will treat both the privacy peers and the server as passive adversaries. This is reasonable in many situations where the privacy peers enjoy a higher level of trust already endorsed in real-life (e.g. a union leader), or restricted by legislation or other factors (hospitals, research institutes, etc.), so that it is unlikely that they deviate from the protocol or try to bias the computation. This does not hold, however, when the privacy peers are from general user population. In this case we can use a random sampling method to verify the computation of privacy peers. Again the P4P architecture still uses the server to defend against external attacks but relies on redundancy to handle active cheating of privacy peers. We leave careful design and analysis of such mechanism as future work.

1.3.1 The Computation

Let ϕ be a small integer (e.g. 32- or 64-bit). Since we need signed values, we consider the specific coset representatives of the integers $\mod \phi$ in the range $-\lfloor \phi/2 \rfloor, \ldots, \lfloor \phi/2 \rfloor$ if ϕ odd, or $-\lfloor \phi/2 \rfloor, \ldots, \lfloor \phi/2 \rfloor - 1$ if ϕ even. We write $\mathbb{Z}_\phi$ for this additive group. [2]

Let a_i be private user data for user i and A be public information. Both can be matrices of arbitrary dimensions with elements from arbitrary domains. Our scheme supports any iterative algorithms whose $(t+1)$-th update can be expressed as

$$A^{(t+1)} = F(\sum_{i=1}^{n} d_i^{(t)}, A^{(t)})$$

where $d_i^{(t)} = G(a_i, A^{(t)}) \in \mathbb{Z}_\phi^m$ is an m-dimensional data vector for user i computed locally, and $A^{(t+1)}$ is a successor iterate to $A^{(t)}$. Typical values for both m and n can range from thousands to millions. Both functions F and G are in general non-linear.

As mentioned before, this is a powerful model that includes a large number of popular data mining and machine learning algorithms. Additional examples include SVD, k-means, ID3, etc., most gradient-based and EM algorithms, and all the algorithms in the statistical query model [75]. [29] showed how to turn these algorithms into such vector addition form (so they can be implemented using Google's MapReduce parallel processing framework [39]). Also see [21, 22, 48] for more examples.

If $d_i^{(t)}$ are computed locally by each user, our protocol allows calculation of the sum, and thence the next iterate $A^{(t+1)}$ without disclosing any information about $d_i^{(t)}$ or a_i. In the following we only describe the protocol for one such iteration since the entire algorithm is simply a sequential invocations of the same protocol. The superscript is thus dropped from the notation.

[2]Since our computation involves addition only, there is no requirement that this group be a field. So ϕ need not be prime, and indeed it simplifies computation to take $\phi = 2^{32}$ or 2^{64} i.e. word-length or long integers.

1.3.2 Leakage by the Sums

Our private summation protocol guarantees that no more information beyond the sums is revealed. One thus must be careful about the potential leakage caused by the sums. For the examples shown in chapters 3, 4 and 5 (the SVD, link analysis and association rule mining) and others such as [109, 21], we are able to show the sums can be approximated from the final result so they do not leak more information at all.

There is ongoing work to provide a more general analysis and we have already obtained some initial results. [3] Basically, using the results in statistical database privacy [11, 43, 49], we are able prove that, under some conditions, the sums are in fact safe. Roughly speaking, these works showed that a noisy version of the sums could guarantee strong privacy provided that Tm is sub-linear in n, where T is the total number of iterations. In our settings, we are able to prove that, in some situations, external noise is *not* necessary for maintaining privacy. Instead, the randomness associated with an adversary's inherent uncertainty about unknown data is enough to prevent it from gaining significant information by seeing the aggregates. We can show that this protection is at least *equivalent* to that provided by the perturbation-based approach in [11, 43, 49].

1.3.3 Security Properties

Since all user vectors are hidden, it is necessary to impose checkable bounds on the user data. Otherwise a single malicious user could corrupt the computation with values as large as $\lfloor \phi/2 \rfloor$ and not be discovered. For this purpose, we use a bound on the L2-norm of each user vector. This is both computationally natural for many applications, and also supports a very efficient, randomized check. The maximum L2-norm for a user vector is defined to be L, which implies that every component of the user vector must be in the range $[-L, L]$.

Note that L must be substantially less than ϕ. First of all, since n user vectors are added to reach a final total mod ϕ, each component value should be less than $1/n$ times

[3]Since they are still preliminary we did not include them in this work.

$\phi/2$. Secondly, L should be much smaller than ϕ to ensure low probability of modular arithmetic anomalies (this is made precise in theorem 1).

Our protocol achieves the following which are similar to those introduced in [21]:

1. **Privacy**: For any honest user i who follows the protocol and inputs valid data, no participants, except herself, should gain any information about d_i, except what is implied by the final aggregate and the validity property below.
2. **Validity**: A user vector d_4 that is included in the computation must satisfy that, with high probability, $|d_i|_2 < L$ where $|d_i|_2$ denotes the L2-norm of the vector d_i.
3. **Correctness**: The computation should produce the correct sum of all *valid* users data.

We adopt the privacy definition of [60] (Chapter 7) and prove the privacy of our protocol in a simulation paradigm that is common in numerous cryptography works and can be traced back to the notion of zero-knowledge [63]. Informally, a protocol is private if, for any adversary that corrupts a subset of the participants as allowed by the protocol, there exists a feasible simulator that, given the corrupted parties data and the final result, can generate a view that, to the adversary, is indistinguishable from the transcript of a real execution of the protocol. This guarantees that whatever information the adversary can obtain after attacking the protocol can be actually generated by himself (by running the simulator) thus no more information about honest parties data is leaked. For formal definition please see [60] (Chapter 7).

The privacy our protocol achieves is information-theoretic, i.e. it holds against an adversary with unbounded computation power, as is provided by the secret sharing and Pedersen commitment scheme we use [99, 33]. However, the success of the validity check relies on some standard assumptions (e.g. DDH or discrete log) so is only computational.

For validity, we provide an extremely efficient zero-knowledge protocol that, instead of verifying $|d_i|_2$ directly, checks the square sum of the vector's projections on some random directions. We show in theorem 1 that this check is effective in bounding $|d_i|_2$. The protocol is probabilistic and has a small failure probability. In a particular failure mode (i.e. false rejection), the protocol leaks one bit of information about user data, i.e. at least one of the projections is large (but it does not allow one to infer which direction(s)). This is made precise in theorem 2.

1.3.4 The Main P4P Computation

Let Q be the initial set of qualified users. Let T_1 and T_2 denote the server and the privacy peer, respectively. The basic computation is carried out as follows:

1. User $i \in Q$ generates a uniformly random vector $u_i \in \mathbb{Z}_\phi^m$ and computes $v_i = d_i - u_i \mod \phi$. She sends u_i to T_1 and v_i to T_2.
2. User i gives a ZK proof to both servers that her input is valid using the protocol that will be described in Section 2.1.1. If she fails to do so, both servers exclude her from Q.
3. If enough (e.g. more than 80% of all users in the group) inputs are collected and pass the validation test, T_1 computes $\mu = \sum_{i \in Q} u_i \mod \phi$ and T_2 computes $\nu = \sum_{i \in Q} v_i \mod \phi$. T_2 sends ν to T_1.
4. T_1 publishes $F(\mu + \nu \mod \phi, A)$ and updates A.

It is straightforward to verify that if both servers follow the protocol, then the final result $\mu + \nu \mod \phi$ is indeed the sum of the user data vectors $\mod \phi$. This result will be correct if every user's vector lies in the specified bounds for L2-norm, which implies that the sum over the integers is the same as the sum $\mod \phi$. Appropriate constraints on L will be given in the statement of theorem 1. Privacy of the computation protocol is summarized, together with that of the verification protocol introduced in section 2.1.1, in theorem 2.

Chapter 2

Zero-Knowledge Tools for Privacy-Preserving Distributed Data Mining

This chapter introduces a number of highly efficient zero-knowledge tools within the P4P framework that can be used to verify the properties of user data such as equality and boundedness. These tools provide practical mechanisms to deal with cheating users and limit their influence on the computation. All of them use a linear number of inexpensive small field operations, and only a constant or logarithmic number of large-field (1024 bits or more) cryptographic operations. They achieve orders of magnitude reduction in running time over standard techniques (from hours to seconds) thus are suitable for large-scale problems.

2.1 L2-norm Boundedness ZKP

Bounding the L2-norm of a user's vector is a natural and effective way to restrict the amount malicious influence on the computation a cheating user could cause. This can be shown from several perspectives. Firstly, notice that the result of the computation depends on the sums of n vectors. To drive the sums away from correct positions by a large amount, a malicious user must input a vector with sufficient "length", which is naturally measured by its L2-norm. This is especially evident for algorithms whose results are simply the vector sums (e.g. k-means). In this case even the precision of the final result is often measured by the L2-norm of the error vector (see e.g. [11]), which, by triangle inequality, is bounded by the sum of the L2-norms of all noise vectors.

Secondly, many perturbation theories measure the perturbation to the system in terms of various forms of (matrix and vector) norms, many of which can be easily transformed into vector L2-morns. For example, let $\tilde{\cdot}$ denote the perturbed quantity and σ_i the i-th singular value of a matrix A, the classical Weyl and Mirsky theorems [118] bound the perturbation to A's singular values in terms of the spectral norm $\|\cdot\|_2$ and the Frobenius norm $\|\cdot\|_F$ of $E := A - \tilde{A}$, respectively:

$$\max_i |\tilde{\sigma}_i - \sigma_i| \leq \|E\|_2 \text{ and } \sqrt{\sum_i (\tilde{\sigma}_i - \sigma_i)^2} \leq \|E\|_F$$

The spectral norm can be bounded from above by Frobenius norm: $\|E|_2 \leq \|E\|_F$. And if each row, denoted a_i, of the matrix A is held by a user, the Frobenius norm of the matrix E can be expressed in terms of vector L2-norms:

$$\|E\|_F = \sqrt{\sum_{i=1}^{n} |\tilde{a}_i - a_i|^2}$$

Clearly bounding the vector L2-norm provides an effective way to bound the perturbation of the results. Similar techniques was also used in e.g. [48].

And finally, bounding the L2-norm can also be the basis of other, more specific checks. For instance, in a voting application, the protocol can be used with $L = 1$ to ensure that each user only exercises one vote.

2.1.1 The Protocol

Overview

A straightforward way of checking the L2-norm of user vector is given in [21] which works with each element and requires $O(m)$ public key operations and ZKPs. We present a novel protocol that uses only constant or $O(\log m)$ such expensive operations thus is orders of magnitude more efficient. The key technique is that, instead of checking each elements, we check the *projections* of the user vector on some random directions. We show that some statistical properties of these projections are related to the L2-norm of the original vector. Therefore by verifying the square sum of a *small* number of such projections (in zero-knowledge), we can check if the L2-norm of a vector with a large number of elements is within a desired bound. The overhead to compute the projections is $O(m)$ but these steps only consist of arithmetic operations in the *small* field. As we will show in our experiments, the cost of such operations is very small compared to the crypto operations (It is not noticeable when $m \leq 10^5$, and is fraction of a second when m reaches 10^6).

Tools

The verification protocol requires some standard primitives for homomorphic computation. These have appeared elsewhere, see e.g. [33], [21] and we summarize only their key properties here. All values used in these primitives lie in the multiplicative group $\mathbb{Z}_q^*$, or in the additive group of exponents for this group, where q is a 1024 or 2048-bit prime. They rely on El-Gamal, RSA or discrete log functions for cryptographic protection of information.

Homomorphic commitment Given an integer value a, a homomorphic commitment to a with randomness r is written $\mathcal{C}(a, r)$. It is homomorphic in the sense that $\mathcal{C}(a, r)\mathcal{C}(b, s) = \mathcal{C}(a + b, r + s)$. It is cryptographically hard to determine a given $\mathcal{C}(a, r)$. We say that a prover "opens" the commitment if it reveals a and r.

ZKP of knowledge A prover who knows a and r (i.e. who knows how to open $\mathcal{A} =$

$\mathcal{C}(a,r)$) can demonstrate that it has this knowledge to a verifier who knows only the commitment $\mathcal{A}$. The proof reveals nothing about a or r.

ZKP for equivalence Let $\mathcal{A} = \mathcal{C}(a,r)$ and $\mathcal{B} = \mathcal{C}(a,s)$ be two commitments to the same value a. A prover who knows how to open $\mathcal{A}$ and $\mathcal{B}$ can demonstrate to a verifier in zero knowledge that they commit to the same value.

ZKP for product Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be commitments to a, b, c respectively, where $c = ab$. A prover who knows how to open $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ can prove in zero knowledge to a verifier who has only the commitments, that the relationship $c = ab$ holds among the values they commit to.

Bit commitment Let $\mathcal{A} = \mathcal{C}(a,r)$ be a commitment to a value a where $a \in \{0,1\}$, which is called a bit commitment. A prover who knows how to open $\mathcal{A}$ can prove in zero knowledge that it commits to either 0 or 1 (but not which).

ZKP for boundedness Let $\mathcal{A} = \mathcal{C}(a,r)$ be a commitment to a value a. Using the above methods, a prover can show that $\mathcal{A}$ contains a k-bit integer, i.e. that it encodes the same value as $\mathcal{B}_{k-1} \cdots \mathcal{B}_0$, where each $\mathcal{B}_j$ encodes 0 or 2^j. If the leading "bit" $\mathcal{B}_{k-1}$ instead encodes 0 or $L - 2^{k-1} + 1$ where $k = \lfloor \log_2 L \rfloor$, then the ZKP proves that $a \in [0, \ldots, L]$ for any k-bit positive L. Adding an additional bit which encodes 0 or $-L$ gives a proof of boundedness in the range $[-L, \ldots, L]$.

Protocol UDVP (User Data Verification Protocol)

Let N be a positive integer which determines the number of challenges, and sets the statistical precision of the verification. The protocol is carried out between each user and the two servers. The execution will be identical for each user so we drop the user index in the notation.

1. **Setup**: After all users send their data to all servers, T_1 broadcasts a random number r to T_2 and all users. Using a public PRG (pseudo-random generator) and r as the random seed, all players generate N independent m-dimensional challenge vector

$c_k \in \{-1, 0, 1\}^m$ with each of its elements generated with IID probabilities $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$, for $k = 1, \ldots, N$.

2. **Projection & Commitment**: For $k = 1, \ldots, N$, the user computes $x_k = c_k \cdot u \mod \phi$, $y_k = c_k \cdot v \mod \phi$, and $s_k = c_k \cdot (u + v) \mod \phi$. Let $s_k = x_k + y_k + b_k$ over the integers, then b_k is either zero or $\pm\phi$. The user computes commitments $\mathcal{X}_k$ to x_k, $\mathcal{Y}_k$ to y_k, $\mathcal{S}_k$ to s_k, $\mathcal{B}_k$ to b_k and finally a commitment $\mathcal{Z}_k$ to the squared sum $z_k = s_k^2$ (computed over the large field $\mathbb{Z}_q$). The user sends all $5N$ commitments to T_1 and T_2.

3. **Consistency Check**: T_1 and T_2 exchange these values to confirm they received identical data from the user. If they do not match, the user's data is rejected.

4. **Commitments Verification**: The user opens $\mathcal{X}_k$ for T_1, and $\mathcal{Y}_k$ for T_2, for $k = 1, \ldots, N$. Both servers confirm that the openings match their data, i.e. T_1 confirms that $\mathcal{X}_k$ is a commitment to x_k and T_2 confirms that $\mathcal{Y}_k$ is a commitment to y_k. The servers communicate the results to each other. If either opening fails or if the user failed to send a complete response to the challenge vector, this user's input is rejected.

5. **Equivalence ZKPs**: For each k, the user proves in zero knowledge to both servers that $\mathcal{S}_k$ encodes the same value as $\mathcal{X}_k\mathcal{Y}_k\mathcal{B}_k$. The user then proves in zero knowledge that $\mathcal{B}_k$ encodes 0 or $\pm\phi$. Finally the user gives a product ZKP to the servers that $\mathcal{Z}_k$ encodes the square of the value that $\mathcal{S}_k$ encodes. If any of these proofs fail, the user's input is rejected.

6. **Boundedness ZKP**: The servers computes the product $\mathcal{Z} = \prod_{k=1}^{N} \mathcal{Z}_k$. The user then provides a ZKP that $\mathcal{Z}$ encodes a value in the range $[0, NL^2/2]$. If this proof succeeds, the user's input is accepted and added to the total.

All the ZKPs used should be implemented non-interactively using the Fiat-Shamir heuristic so that the users can upload the data to the servers in a batch without further interaction.

Field/Group Sizes

The protocol assumes that the size of the cryptographic field $\mathbb{Z}_q$ used for commitments and ZKPs is much larger than the "small" group $\mathbb{Z}_\phi$ used for additive secret-sharing. A transition happens when z_k is computed from s_k. The value of s_k lies in the small group, while $z_k = s_k^2$ is computed in the large field. The sum $z = \sum_{i=1}^{n} z_k$ should be less than q to avoid modular reduction of z in the large field. This will almost surely be true. Since ϕ is typically 64 bits or less, z_k will have at most 128 bits, while z will be at most $128 + \log_2 n$ which is much less than 1024.

Theorem 1. *Let $|d|_2$ denote the L2-norm of user vector d, and L be the specified bound on this norm. Define $\delta = L^2/|d|_2^2$. Then if $|d|_2 < L$, and further $\delta > 2$, the probability that a user vector is (incorrectly) rejected is at most:*

$$Pr[z > NL^2/2] \leq \left(\tfrac{\delta}{2}\exp(1 - \tfrac{\delta}{2})\right)^N$$

If instead $|d|_2 > L$, the probability that a user vector is (incorrectly) accepted is at most:

$$Pr[z < NL^2/2] \leq \left(\left(\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2\right)\exp\left(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2\right)\right)^N$$

Furthermore, these bounds are valid using modular arithmetic as per the above protocol if L satisfies:

$$L \leq \phi/\max(56.5\sqrt{m}, 2n)$$

where n is the number of users and m is the vector dimension.

Proof. We first present proofs for the tail bounds assuming the total s_k on each round exactly represents the weighted sum of user vectors $s_k = \sum_{j=1}^{m} c_k \cdot d$. Because the sums are actually computed $\mod \phi$, s_k may differ by a multiple of ϕ from the total over the integers. We deal with modular arithmetic effects later in this section.

Statement Let $z_k = s_k^2$, and $z = \sum_{k=1}^{N} z_k$. Let $V = E[z_k]$ for $k = 1, \ldots, N$ and $\delta = (L/|d|_2)^2 = L^2/(2V)$. Then the probability that a user input fails the test is probability that $Pr[z > NL^2/2] = Pr[z > \delta NV]$, and we claim that

$$Pr[z > \delta NV] \leq \left(\tfrac{\delta}{2}\exp(1 - \tfrac{\delta}{2})\right)^N \qquad \text{where } \delta > 2$$

Conversely, with the same definitions and if $\delta < 1$, the user will pass the test if $Pr[z < NL^2/2] = Pr[z < \delta NV]$, which has a bound

$$Pr[z < \delta NV] \leq \left(\left(\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2\right) \exp\left(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2\right)\right)^N$$

where $0 < \delta < 1$

Proof

Since s_k is a sum of independent random variables, the pdf of s_k typically has gaussian-decay tails, but the squares $z_k = s_k^2$ in general are not gaussian and have simple exponential tails. We can still prove Chernoff-style bounds for the tails of z that show exponential decrease in the number of trials, but the bounds have only simple-exponential decrease away from the mean. In all cases, we use bounds of the moments of z_k which are derived later in Lemma 1.

Upper Tail

First, for the upper tail let $\delta > 1$, and since $E[z] = NV$, we evaluate

$$Pr[z > \delta NV] = Pr[\exp(tz) > \exp(t\delta NV)]$$

and applying a Markov bound we obtain

$$Pr[z > \delta NV] \leq \frac{E[\exp(tz)]}{\exp(\delta NV)} \tag{2.1}$$

and since the z_k are independent for $k = 1, \ldots, N$, we can factor the expected value as the product of $E[\exp(tz_k)]$. Since we have all the moments of z_k, we can compute this value as a power series:

$$\begin{aligned} E[\exp(tz_k)] &= \sum_{i=0}^{\infty} t^i E[z_k^i]/(i!) \\ &\leq \sum_{i=0}^{\infty} (tV/2)^i (2i)!/(i!)^2 \\ &= \sum_{i=0}^{\infty} (tV/2)^i \binom{2i}{i} \end{aligned}$$

and since $\binom{2i}{i} \leq 4^i$, this series will converge so long as $2tV < 1$. The series is then geometric, and has a bound of:

$$E[\exp(tz_k)] \leq \frac{1}{1-2tV}$$

and substituting into (2.1) gives

$$Pr[z > \delta NV] \leq \frac{1}{(1-2tV)^N \exp(t\delta NV)} \tag{2.2}$$

and this bound is optimized by maximizing $(1-2tV)\exp(t\delta V)$. Taking derivatives and solving gives $t = 1/(2V) - 1/(V\delta)$. The bound is valid so long as $0 < t < 1/(2V)$, which is true if $\delta > 2$. Substituting, we obtain:

$$Pr[z > \delta NV] \leq \left(\tfrac{\delta}{2}\exp(1-\tfrac{\delta}{2})\right)^N \qquad \text{where } \delta > 2 \tag{2.3}$$

Lower Tail

Now let $\delta > 0$, $E[z] = NV$, we evaluate

$$Pr[z < \delta NV] = Pr[\exp(-tz) > \exp(-t\delta NV)]$$

for $t > 0$, and applying a Markov bound we obtain

$$Pr[z < \delta NV] \leq \frac{E[\exp(-tz)]}{\exp(-t\delta NV)} \tag{2.4}$$

the expected value factors as before into terms $E[\exp(-tz_k)]$. The expansion is an alternating sum which is difficult to bound, so instead we truncate it using the inequality

$$\exp(-y) \leq 1 - y + y^2/2$$

which holds for all $y > 0$. This gives the bound:

$$\begin{aligned} E[\exp(-tz_k)] &\leq E[1 - tz_k + t^2 z_k^2/2] \\ &= 1 - tV + \frac{t^2}{2}E[z_k^2] \\ &\leq 1 - tV + \frac{3}{2}t^2V^2 \end{aligned} \tag{2.5}$$

where the last step used the moment bounds from Lemma 1. Substituting into (2.4) gives

$$Pr[z < \delta NV] \leq \left((1 - tV + \tfrac{3}{2}t^2V^2)\exp(tV\delta)\right)^N \tag{2.6}$$

Minimizing the RHS involves solving a quadratic equation which is a function of δ. The solution can be approximated as $t \approx (1/2 - 5/12\delta)/V$. We can use this value as a bound in any case, giving:

$$Pr[z < \delta NV] \leq ((\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2)\exp(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2))^N \tag{2.7}$$

Dealing with Modular Arithmetic

In order for the secret shares not to leak information about user data, modular arithmetic is used. We use the notation $x[i]$ for the i^{th} component of the vector x. We denote by $\bar{s}_k = \sum_{j=1}^m c_k[j]d[j]$ the sum over the integers, and by s_k this sum reduced $\mod \phi$, which is what the protocol actually computes. Then we have

$$\bar{s}_k = s_k + w\phi$$

for some integer w. The modular arithmetic provides additional ways for the user to cheat. e.g. the user might set some components of her vector to $\phi/2$. If an even number of those are included in the checksum, they will be removed by the modular arithmetic, leading to a small s_k. However, we show now that any such "large" components will cause the protocol to fail almost surely. We consider the following two cases:

1. All components $d[i]$ of the user's vector are in the range $[-4L, 4L]$
2. Some component $d[i]$ has magnitude larger than $4L$.

Note that case 1 includes both legal and illegal inputs, since the largest legal magnitude for any component is L. Case 2 vectors have overall magnitude greater than L and are strictly illegal.

Case 1: If all components of the user vector are in the range $[-4L, 4L]$, then the maximum variance of this vector is $V = 32mL^2$. The reduced s_k will be equal to $\bar{s}_k$ as long as $\bar{s}_k$ is in the range $\mathbb{Z}_\phi$, i.e. as long as $|\bar{s}_k| \leq \phi/2$. By setting δ to the ratio of squared limit over variance $\delta \geq \phi^2/(32mL^2)$, and $N = 1$ we can use the upper tail bounds computed earlier to bound a single s_k.

$$Pr[|\bar{s}_k| > \phi/2] = Pr[z_k > \delta V] \leq \left(\tfrac{\delta}{2}\exp(1 - \tfrac{\delta}{2})\right)$$

A typical safe value would be $\delta = 100$, giving a failure probability of 2.6×10^{-20}. The bound L must satisfy $L \leq \phi/\sqrt{32\delta m}$, which for $\delta = 100$ becomes $L \leq \phi/(56.5\sqrt{m})$. This constraint would normally be satisfied in any practical system, because L must be small enough to allow $\bar{s}_k$ totals to be computed without wrapping $\mod \phi$. That is if there are n users, the bound L should be such that $nL \leq \phi/2$, because a legal user input may have a value of L in one element only. Satisfying both constraints gives us the result: $L \leq \min(\phi/(56.5\sqrt{m}), \phi/(2n))$ or $L \leq \phi/\max(56.5\sqrt{m}, 2n)$

Case 2: Some $|d[i]| > 4L$. Fix this i, and let $\bar{s}_{-i}$ denote the sum $\sum c[j]d[j]$ of all terms $j \neq i$ over the integers. Now either $\bar{s}_{-i}$ is in some range $[-2L, 2L] + k\phi$ or it isnt (we say it is "legal" it is is in such a range). The final total $\bar{s} = c[i]d[i] + \bar{s}_{-i}$ differs from $\bar{s}_{-i}$ by either 0 or $d[i]$ where $4L \leq |d[i]| \leq \phi/2$. If $\bar{s}_{-i}$ is legal, then $\bar{s}_{-i} \pm d[i]$ must be illegal, which has probability 1/2. If $\bar{s}_{-i}$ is illegal to begin with, then at most both the offsets $\pm d[i]$ will be legal, which again has probability 1/2. If p is the probability that $\bar{s}_{-i}$ is legal at first, the probability that $\bar{s}$ is legal is at most $\frac{1}{2}p + \frac{1}{2}(1-p) = \frac{1}{2}$.

Now let $q \leq N$ be the number of challenges for which $\bar{s}_k$ is illegal, i.e. the number of k for which $\bar{s}_k > 2L$. For each of these $z_k > 4L^2$ and the total z will be at least $4qL^2$. The overall user data verification will (incorrectly) succeed if $z < NL^2/2$, which can only happen if $q < N/8$. The probability that this happens is the tail of a Bernoulli distribution over uniform trials with probability $\geq \frac{1}{2}$. Using standard formulae [89], this probability is bounded by:

$$Pr[z < NL^2/2] \leq 0.8173^N$$

This probability is strictly less than the lower tail bound derived above which is never better than $\frac{7}{8}^N = 0.875^N$. So the latter bound dominates, and we do not separately quote the probability for modular wrap-around error.

Lemma 1. *For independent random variables $c[j]$ in $\{-1, 0, 1\}$ with probabilities $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$ respectively, and let $s = \sum_{j=1}^{m} c[j]d[j]$, and $z = s^2$. Then all positive moments of z satisfy:*

$$E[z^q] \leq \frac{(2q)!}{q!2^q}V^q = (1 \cdot 3 \cdot 5 \cdots (2q-1))V^q \leq (qV)^q$$

where $V = E[z] = E[s^2]$ as before.

Proof

We rewrite the sum for each moment as: $E[z^q] = E[s^{2q}] = E[(\sum_{i=1}^{m} s[i])^{2q}]$ and fully expanding the last term gives:

$$E[z^q] \;=\; \sum_{r=1}^{2q} \sum_{i_1+\cdots+i_r=2q}^{1\leq i_1,\ldots,i_r\leq 2q} \binom{2q}{i_1, i_2, \ldots, i_r} E[s[j_1]^{i_1} \cdots s[j_r]^{i_r}] \tag{2.8}$$

where $1 \leq j_1 < j_2 < \cdots < j_r \leq m$. Next we notice that each $s[j]$ is symmetric: $Pr[s[j] = v] = Pr[s[j] = -v]$. So every term containing an odd power of some $s[j]$ has expected value zero. wlog we can assume that every index $i_1, \ldots, i_r$ in the expression above is even.

The expected values in the last formula can be computed directly since the s_j are independent:

$$E[s[j_1]^{2i_1} \cdots s[j_r]^{2i_r}] = \frac{1}{2^r} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r}$$

Rewriting (2.8) using this expansion, and using only even powers gives:

$E[z^q] =$

$$\sum_{r=1}^{2q} \sum_{i_1+\cdots+i_r=q}^{1\leq i_1,\ldots,i_r\leq q} \binom{2q}{2i_1, 2i_2, \ldots, 2i_r} 2^{(-r)} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r} \tag{2.9}$$

In order to simplify this last expression, we consider the expansion of $(2V)^q$ which is:

$(d[1]^2 + \cdots + d[m]^2)^q =$

$$\sum_{r=1}^{q} \sum_{i_1+\cdots+i_r=q}^{1\leq i_1,\ldots,i_r\leq q} \binom{q}{i_1, i_2, \ldots, i_r} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r} \tag{2.10}$$

which contains exactly the same products of $d[j]$'s. We take the ratio of the coefficients of $d[j_1]^{2i_1} \cdots d[j_r]^{2i_r}$ in (2.9) and (2.10), giving

$$R = 2^{-r} \binom{2q}{2i_1, \ldots, 2i_r} / \binom{q}{i_1, \ldots, i_r} \tag{2.11}$$

We expand this first as

$$\frac{(2q)!}{q!} \frac{i_1!}{(2i_1)!} \cdots \frac{i_1!}{(2i_r)!} 2^{-r}$$

and notice that $\frac{i_j!}{(2i_j)!} \leq 2^{-i_j}/(i_j)!$. Making these substitutions gives

$$R \leq \frac{(2q)!}{q!} \frac{1}{2^{i_1} i_1!} \cdots \frac{1}{2^{i_r} i_r!} 2^{-r} = (q)^q \frac{1}{i_1! \cdots i_r! 2^r}$$

and finally it is easy to show that $i_1! \cdots i_r! 2^r \geq 2^q$. This can be done inductively by starting with $r = q$, and all $i_j = 1$, and "walking" to any desired partition $i_1 + \cdots + i_r$ of q, each step merging some i_j which is $= 1$ with another. The number of groups r decreases by 1 at each step which reduces 2^r by 2, but some i_l is incremented at the same time, and so $i_l!$ is multiplied by *at least* 2. Substituting for these expressions in the denominator of R in the last equation gives:

$$R \leq \frac{(2q)!}{q!} 4^{-q}$$

Now if we multiply equation (2.10) by this value, we guarantee that every term in its expansion is at least as great as the coefficient in equation (2.9). Or in other words,

$$E[z^q] \ \leq \frac{(2q)!}{q! 2^q} V^q = (1 \cdot 3 \cdot 5 \cdots (2q-1)) V^q \leq (qV)^q \qquad \text{QED} \tag{2.12}$$

$\square$

The first (false rejection) bound is quite steep. A drop-off of 2^{-N} is achieved for $\delta \approx 5.3566$. If $\delta = 4$ (user vector norm is one half of L), then the roll-off is 0.736^N. The second (false acceptance) bound is considerably shallower. In the limit as $\delta \to 0$, the bound is $\frac{7}{8}^N$. For $\delta = 0.25$ (user vector norm is twice L), the bound is 0.9265^N, while for $\delta = 0.5$, it is 0.9672^N.

Theorem 2. *The computation is private. Furthermore, for the purpose of verifying $z < NL^2/2$, UDVP is zero-knowledge. When used to bound $|d|_2$, only in the event of false* *rejection does it reveal information about valid user data other than its validity. And this leakage is exactly the fact that $z > NL^2/2$.*

The proof of theorem 2 consists of standard simulation of a secret sharing scheme and straightforward invocation of sequential composition ZKPs theorem [62], and is omitted.

Note that what the protocol actually verifies is whether $z < NL^2/2$. For this purpose it is zero-knowledge in that the verifier learns nothing except the fact when it is true. However, this assertion is not completely equivalent to $|d|_2 < L$ and theorem 1 quantifies its effectiveness in bounding $|d|_2$. There is a small probability of false rejection where $|d|_2 < L$ but $z > NL^2/2$. In this case the verifier learns that at least one of the projections is large. This does not violate the zero-knowledge property of the ZKP with regard to $z < NL^2/2$ because in that context this is treated as a failed proof and there is no need to protect this information. In contrast, this is considered a leakage in our system since the user data is actually valid (thus should be protected). We believe such leakage is acceptable in most applications because (1) the false rejection probability can be reduced exponentially by using large N and (2) the leakage is very small: the verifier only learns no more than the fact that at least one of the projections is large. In particular it does not learn *which* projection is large.

COMPLEXITY The protocol computes $O(N)$ commitments and square ZKPs. In addition, the boundedness ZKP at the last step involves $O(\log(NL))$ steps. As theorem 1 shows, the failure probability is exponentially decreasing with N. In practice a small constant N (e.g. 50) can provide fairly good guarantee. So the total cost caused by the expensive large integer operations is $O(\log(L))$. In most cases L is either constant or polynomial in m (recall that L is the bound on the L-2 norm of an m-dimensional vector). Therefore the number of large integer operations is bounded by $O(\log m)$. This is far superior to using standard techniques which requires $O(m)$ such operations.

2.1.2 Simulations of Typical Behavior

The bounds we derived earlier show that *any* user vector whose L2-norm is substantially below L will almost surely be accepted, while any vector that is substantially above will surely be rejected. In terms of actual behavior however, the tail bounds we derived may not be very tight. Here we present some simulations for typical user data to show what behavior would be expected. Simulation could also potentially be useful to honest or dishonest users: in either case, a user with an actual input vector d_i can determine through simulation the probability of that value being accepted by the server. We choose 3 specific cases:

1. Random uniform values: every component $d[j]$ of the user vector is drawn from the same uniform distribution.
2. Zipf distribution: component $d[j]$ has value proportional to $1/j$.
3. Single element: only one value in in the user vector is non-zero.

In all cases, user vectors are normalized so their L2-norm is fixed at some value V_d. We will vary this value relative to the threshold L and determine the probability of acceptance.

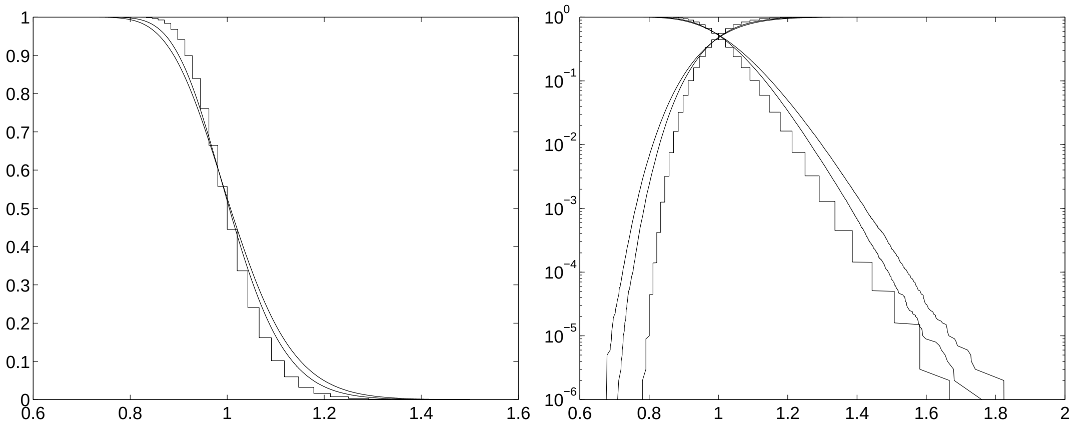

Figure 2.1. (a) Linear and (b) log plots of probability of user input acceptance as a function of V_d/L for $N = 50$. (b) also includes probability of rejection. In each case, the steepest (jagged curve) is the single-value vector (case 3), the middle curve is Zipf vector (case 2) and the shallow curve is uniform vector (case 1)

The first two cases are representative of likely user data, e.g. case 1 could represent ratings for movies while case 2 could represent word counts in email or text messages. The third case is representative of a user who wants to bias the total by using a maximum value for one item. A second reason for these choices is that cases 1 and 3 represent probable extremes of distributions of user vectors. All sums s_k are sums of 3-valued $s_k[j]$. The more terms in this sum and the more similar those terms, the closer will be the final distribution to a gaussian. The s_k produced by case 1 are almost perfectly gaussian. The s_k for Zipf distributed data are mixtures of terms with very different weights, and are "less" gaussian. Finally, the s_k for single-element vectors retain a 3-valued distribution and are very far from gaussian. Any distribution the user can produce will be a sum of such $s_k[j]$, and will probably lie between the extremes of cases 1 and 3.

The simulations used $N = 50$, $m = 100$, and were repeated 10^6 times. Figure (2.1) shows probabilities of acceptance or rejection for the 3 cases as a function of the ratio V_d/L. Increasing N by a factor α should cause the log plots to scale by α in their y-values. When $N = 50$, the upper tail bound from theorem 1 has an asymptotic slope of 25 in $\log(Pr)$ vs. $\log(\delta)$ plots. The lower tail bound slope is significantly shallower because of "saturation" of the probability to $\frac{7}{8}^N$as $\delta \to 0$. The x-axes in figure 1 involve $|d|/L$ which is $1/\sqrt{\delta}$. The expected slopes from the tail bounds would be 12.5 for the rejection probability curve, and less for the acceptance curve. The actual slope observed for rejection is about 50, while it is around 35 for acceptance. So the typical threshold behavior for the probabilistic L2-bound is much sharper than the asymptotic bounds from theorem 1.

2.1.3 Implementation and Evaluation

We have implemented the protocols in Java using a NativeBigInteger implementation from the I2P anonymous network (http://www.i2p.net/). The source code will be made available as a toolkit to the public shortly. We measured the performance on a 2.8GHz Xeon. All tests were carried out using El-Gamal commitments and ZKPs [33] in a large field $\mathbb{Z}_q$ for a 1024-bit prime q. N was 50, and L was either a 40-, 20- or 10-bit number. The basic bit commitment ZKP takes 33.7 ms for the verifier and 57.3 ms for the prover. The square ZKP takes 35.7 ms verifier time and 24.3 ms prover time. Figure 2.2 plots prover (user) and verifier times for the L2-norm validation protocol as a function of the vector size m.

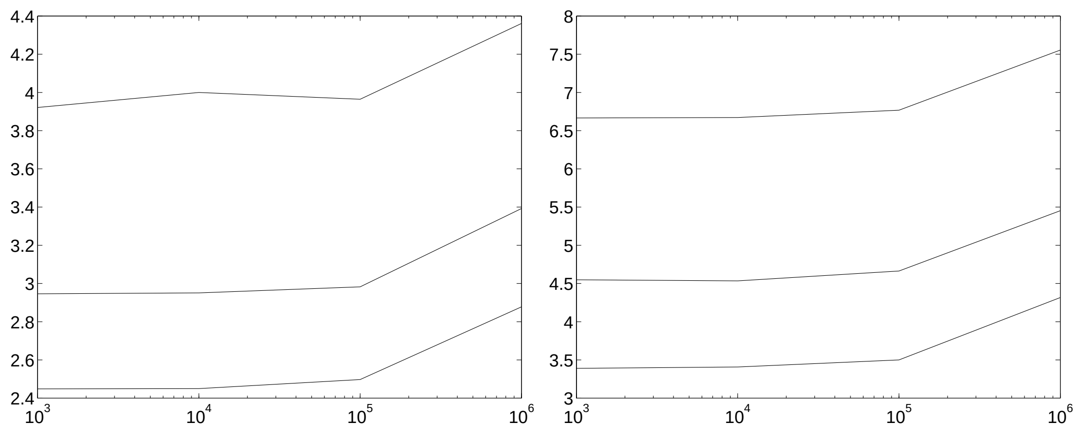

Figure 2.2. (a) Verifier and (b) prover times in seconds for the validation protocol with $N = 50$, where (from top to bottom) L has 40, 20, or 10 bits. The x-axis is the vector length m.

Both were dominated by cryptographic operations in these experiments, even at $m = 10^6$. Other steps, such as random vector generation, or computation of all the products $c_k \cdot d$ by the prover, took a fraction of a second. At $m = 10^6$, verifying one user's vector takes only a few seconds. In contrast, using standard techniques which requires $O(m)$ square ZKPs, as is done in [21], both the verifier and prover time is in hours (about 10 hours for the verifier and 6.7 hours for the prover). Our protocol is orders of magnitudes more efficient. The server can easily process over 15000 users each day with a *single* PC. Since the protocol for each user is independent of each other, the server can support larger user group with a cluster, which is a common practice for today's service providers. Most of the applications such as collaborative filtering require only infrequent update (once per day or more), the performance we obtained is more than enough for them. The communication overhead is also very small since it passes very few large integers. The communication per client is only a few kilobytes, while other solutions require some hundreds of megabytes.

2.2 Zero-knowledge Test of Vector Equivalence

2.2.1 Tools

The equivalence protocols require the following additional cryptographic primitives which are also standard. They have appeared elsewhere, see e.g. [99, 33]. Here we summarize their key properties.

Homomorphic commitment We use Pedersen's discrete log based scheme [99] for it admits an efficiency ZKP for equivalence. Let p and q be two large primes such that $q|p-1$. Let $\mathbb{Z}_p^*$ denote the multiplicative group of integers modulo the prime p. We use G_q to denote the unique subgroup of $\mathbb{Z}_p^*$ of order q. The discrete logarithm problem is assumed to be hard in G_q. Let g and h be two generators of G_q such that $\log_g h$ is unknown to anyone. [1] A commitment to a is computed as $\mathcal{C}(a,r) = g^a h^r \mod p$ where $r \leftarrow_R \mathbb{Z}_q$. From now on, $\mathcal{C}(a,r)$ will denote such Pedersen commitment function. We will omit the randomness r from the notation, and simply write $A = \mathcal{C}(a)$, if it is not necessary to identify it.

Multiply by A Constant Let A be commitment to $a \in \mathbb{Z}_q$ and $c \in \mathbb{Z}_q$ a constant. One can easily obtain a commitment to $d = ac$ by

$$D = A^c \mod p$$

This follows immediately from homomorphism. However, only the prover who knows how to open A can open D.

3-way Commitment and ZKP For integer $c \in \mathbb{Z}_q$, a 3-way commitment, denoted $(0, \pm c)$-commitment, is a commitment to one of $0, c$ or $-c$. For such commitment we can construct an efficient zero-knowledge proof.

Let $a \in \{0, c, -c\}$ be the number to be committed to. The prover computes two

[1] A generator of G_q can be easily found by selecting an element $a \leftarrow_R \mathbb{Z}_p^*, a \neq 1$ and testing if $a^q = 1$, since any element $a \neq 1$ in G_q generates the group. g and h can chosen by the two talliers using a coin-flipping protocol.

commitments B and C such that

$$B = \mathcal{C}(0), C = \mathcal{C}(0) \text{ if } a = 0$$

$$B = \mathcal{C}(1), C = \mathcal{C}(0) \text{ if } a = c$$

$$B = \mathcal{C}(0), C = \mathcal{C}(1) \text{ if } a = -c$$

The prover also provides zero-knowledge proofs that both B and C encode either 0 or 1 using the *bit commitment* proof of [33]. Finally, the commitment to a is simply

$$A = B^c C^{-c} \mod p$$

and the 3-way $(0, \pm c)$-commitment proof consists of (B, C) and their bit commitment proofs.

To verify the proof, a verifier checks that $A = B^c C^{-c} \mod p$ and that the bit commitment proofs are valid. If all these verifications are successful, the verifier accepts the proof. Otherwise it rejects it.

It is easy to show that only when A encodes one of $\{0, c, -c\}$ will the verifier accept the proof. And it is zero-knowledge due to the hiding property of the commitment scheme. The proof can also be made non-interactive by hashing the verifiers response.

2.2.2 ZK Test of Equivalence

In this section we introduce two equality test protocols. One tests a single element, the other the whole user vector. Both enjoy the following:

1. No information about user data is leaked;
2. Only a small number of public key operations are involved;
3. Users do not need to be involved after the initial data input stage of the main P4P protocol.

Equality Test of A Single Element

Let a_i be the element of user i that defines the equivalence relation E which partitions U. Recall that the two shares of a_i, denoted a_{i1} and a_{i2}, are already sent to T_1 and T_2, respectively, in the main P4P protocol. The goal is, given two user indexes i, j, to determine whether $a_i = a_j$.

This task is not as trivial as it appears. It is true that, with homomorphic commitment, verifying whether two commitments contain equal numbers in zero-knowledge is easy provided there is a prover holding the pre-images [99, 33]. Our setting, however, is a different model. Namely, in P4P, there is no such a prover who knows both numbers. Instead, each tallier holds a share of each number. Collaboratively they want to determine whether the two are equal. In other words, ours is not a zero-knowledge proof task. Rather, it is a zero-knowledge test or a two-party computation problem computing a boolean function that returns 1 if $a_i = a_j$ and 0 otherwise. This, of course must be done without leaking any information about the numbers.

The difficulty in applying existing ZKP lies in the fact that the definition of zero-knowledge in a ZKP system protects prover's privacy only when the statement is true. To see this, let $A_i = \mathcal{C}(a_i, r_i)$ and $A_j = \mathcal{C}(a_j, r_j)$ where the prover knows (a_i, r_i, a_j, r_j). The technique for proving A_i and A_j contain the same number involves the prover revealing $\delta = r_i - r_j$ and the verifier checking if $A_i A_j^{-1} = h^{\delta}$ holds [99, 33]. The problem is, if $a_i \neq a_j$, revealing δ also reveals some information about $a_i - a_j$. Namely once the random mask is exposed, one can obtain $g^{a_i - a_j} = A_i A_j^{-1} h^{-\delta} \mod p$. And when $a_i - a_j$ is small, it is easy to recover it. In a standard ZKP setting, the prover can just admit $a_i \neq a_j$ when it is the case, thus avoiding leaking information. But this trick is not possible in P4P where no such prover exists. The same is true for the DISPEP and PEP [87] techniques which use ElGamal encryption.

We develop the following protocol to address this issue. To enable equality test, the users first escrow some information with the talliers. Specifically, for $j = 1, 2$, user i and the talliers perform the following:

1. User i computes $A_{ij} = \mathcal{C}(a_{ij}, r_{ij})$ where $r_{ij} \leftarrow_R \mathbb{Z}_q$. She also prepares a 3-way $(0, \pm\phi)$-commitment B_i with pre-image $b_i = a_i - (a_{i1} + a_{i2})$ and randomness s_i. She then shares both b_i and s_i:

$$s_{i1} \leftarrow_R \mathbb{Z}_q, \quad s_{i2} = s_i - s_{i1} \mod q$$
$$b_{i1} \leftarrow_R \mathbb{Z}_q, \quad b_{i2} = b_i - b_{i1} \mod q$$

   She sends $(A_{ij}, r_{ij}, b_{ij}, s_{ij})$ to T_j and broadcasts to both talliers $B_{ij} = \mathcal{C}(b_{ij}, r_{ij})$, B_i and its corresponding 3-way $(0, \pm\phi)$-commitment proof.

2. T_j verifies that $A_{ij} = \mathcal{C}(a_{ij}, r_{ij})$ and $B_{ij} = \mathcal{C}(b_{ij}, r_{ij})$. Both talliers verify that $B_i = B_{i1}B_{i2}$ and that the $(0, \pm\phi)$-commitment proof is valid. If any of the verification fails, user i is excluded from the computation.

The above is executed in the user input stage together with data validation. It is the only stage involving user interaction. The actual test can be carried out between the two tallier afterwards. The users do not have to be online at all times.

EQUALITY-TEST:

Without loss of generality, suppose we want to check if $a_1 = a_2$. In the following description, $j \in \{1, 2\}$.

1. Both talliers compute

$$A_1 = A_{11}A_{12}B_{11}B_{12} \mod p$$
$$A_2 = A_{21}A_{22}B_{21}B_{22} \mod p$$

   and $\Delta = A_1 A_2^{-1} \mod p$.

2. T_j computes $\delta_j = (r_{1j} + s_{1j}) - (r_{2j} + s_{2j}) \mod q$.

3. T_j generates a random number $k_j \leftarrow_R \mathbb{Z}_q^* \setminus \{1\}$ and computes $\Delta_j = \Delta^{k_j} \mod p$, $H_j = h^{k_j} \mod p$.

4. The two talliers exchange (Δ_j, H_j). If T_i finds $H_j = h, j \in \{1,2\}, j \neq i$, he aborts the protocol.

5. T_1 publishes $\bar{H}_1 = (H_1 H_2)^{\delta_1}$ which is $h^{k\delta_1}$, and T_2 publishes $\bar{H}_2 = (H_1 H_2)^{\delta_2}$ which equals to $h^{k\delta_2}$, where $k = k_1 + k_2 \mod q$.

6. Both talliers verify if

$$\Delta_1 \Delta_2 \equiv \bar{H}_1 \bar{H}_2 \pmod p \tag{2.13}$$

If it holds, then $a_1 = a_2$. Otherwise $a_1 \neq a_2$.

Theorem 3. *The above protocol correctly tests if $a_1 = a_2$. Furthermore it does not leak any information about user data.*

Proof. The completeness of the protocol follows the homomorphism property. Note that Δ computed in Step 1 is a commitment to $a_1 - a_2$ with randomness $\delta_1 + \delta_2$. It follows that $\Delta_1 \Delta_2$ in Equation 2.13 is a commitment to $k(a_1 - a_2)$ with randomness $k(\delta_1 + \delta_2)$. If $a_1 = a_2$, $\Delta_1 \Delta_2$ should open to 0, and Equation 2.13 should hold.

The soundness is guaranteed by the binding property of the commitment and the fact that the probability of $k_1 + k_2 \equiv 0 \mod q$ is very small (only $1/q$).

To show that the protocol is zero-knowledge, we construct a simulator that takes as inputs the corrupted player's data, the public information, and the final output, and interacts with the adversary in a simulated execution of the protocol. We need to show that this execution is indistinguishable to the adversary.

Without loss of generality, let us suppose tallier T_1 is corrupted. Note that Δ is common inputs to both talliers computed from users' public commitments. The simulator only needs to produce the rest of the conversation.

Let $\bar{\mathbb{Z}}_q^* = \mathbb{Z}_q^* \setminus \{1\}$. The view of the adversary (i.e. T_1) during a real execution of the protocol is: [2]

$$VIEW^{T_1}_{Real} = [h, k_1, \delta_1, h^{k_2}, h^{(k_1+k_2)\delta_2}]_{k_1,k_2 \leftarrow_R \bar{\mathbb{Z}}^*_q, \delta_1,\delta_2 \leftarrow_R \mathbb{Z}_q}$$

For a non-passing execution (i.e. one that outputs $a_1 \neq a_2$), the simulator just generates two random numbers $k'_2 \in \bar{\mathbb{Z}}^*_q, \delta'_2 \leftarrow_R \mathbb{Z}_q$ and uses k'_2, δ'_2 in place of k_2, δ_2 in the protocol. Clearly the transcript of the simulation follows the same distribution as that of an actual execution of the protocol and only with negligible probability will the simulated protocol (incorrectly) output $a_1 = a_2$.

For a passing test (i.e. one execution that outputs $a_1 = a_2$), the simulator works as follows:

It generates $k'_2 \leftarrow_R \bar{\mathbb{Z}}^*_q$ and computes Δ_2 and H_2 as specified by the protocol but with k'_2 in place of k_2. It then computes the rest of the information as required by the protocol as

$$\begin{aligned} \bar{H}_2 &= \Delta_1 \Delta_2 / \bar{H}_1 \mod p \\ &= h^{(k_1+k'_2)\delta_2} \mod p \end{aligned}$$

The adversary's view in the simulated execution is then

$$VIEW^{T_1}_{Sim} = [h, k_1, \delta_1, h^{k'_2}, h^{(k_1+k'_2)\delta_2}]_{k_1,k'_2 \leftarrow_R \bar{\mathbb{Z}}^*_q, \delta_1,\delta_2 \leftarrow_R \mathbb{Z}_q}$$

Clearly this distribution is identical to $VIEW^{T_1}_{Real}$, the adversary's view in a real execution.

And finally the final output reveals $g^{k(a_1-a_2)}$, not $g^{a_1-a_2}$. When $a_1 \neq a_2$, this quantity leaks no information about either a_1 or a_2, or their difference. This is because when $a_1 \neq a_2$, the difference is in the multiplicative group $\mathbb{Z}^*_q$ and has an inverse mod q. For any given value c and a_1, a_2, there is a $k = c(a_1 - a_2)^{-1} \mod q$ such that $k(a_1 - a_2) \equiv c \mod q$. In other words, $a_1 - a_2$ is equally likely to take any values in $\mathbb{Z}^*_q$ even $k(a_1 - a_2)$ is revealed. $\square$

The use of B_j in the protocol is to deal with modular reduction. Note that a_i, a_{i1} and a_{i2} are all in the small field $\mathbb{Z}_\phi$. In order for the shares a_{i1} and a_{i2} not to leak any

[2] We omit the public commitments since they are uniformly randomly distributed in G_q (recall that Pedersen's commitment scheme is information-theoretic hiding [99]).

information about a_i, they should be computed as $a_{i1} \leftarrow_R \mathbb{Z}_\phi, a_{i2} = a_i - a_{i1} \mod \phi$. This means $b_i = a_i - (a_{i1} + a_{i1})$ can be 0 or $\pm\phi$. Using B_j in the protocol is to correct the modular reduction and obtain the actual commitment to a_i. Also note that the sharing of b_i is in the big field $\mathbb{Z}_q$ and there is no modular reduction problem here because of commitment is in G_q, the cyclic group of order q.

Equality Test of the Whole Vector

Equality test of the whole vector can be done via m element tests introduced in Section 2.2.2. However, this involves $O(m)$ public key operations and is not practical for large m. The following protocol, in contrast, requires only $O(1)$ public key operation and is very efficient.

Our protocol uses similar ideas as in the L2-norm boundedness ZKP, i.e. instead of checking every element, it checks the projections of the vectors on a random challenge vector. We show later that, if two vectors have equal projections on these directions then, with high probability, the two vectors are equal.

Suppose we are checking if $d_1 = d_2$. Recall that after the input and validation stage, T_1 holds u_1, u_2 and T_2 has v_1, v_2 such that $d_1 = u_1 + v_1 \mod \phi$ and $d_2 = u_2 + v_2 \mod \phi$.

EQUALITY-TEST-V:

1. T_1 and T_2 generate a random challenge vectors $c \leftarrow_R \mathbb{Z}_q^m$ using the a common random seed r they agreed upon with some protocol.
2. T_1 computes $x = c \cdot (u_1 - u_2) \mod \phi$ and T_2 computes $y = c \cdot (v_1 - v_2) \mod \phi$.
3. T_1 commits to x with $X = \mathcal{C}(x, \delta_1), \delta_1 \leftarrow_R \mathbb{Z}_q$. Similarly T_2 commits to y: $Y = \mathcal{C}(y, \delta_2), \delta_2 \leftarrow_R \mathbb{Z}_q$. The two exchange X, Y and compute the following 3 numbers:

$$\begin{aligned} Z_1 &= XY \mod p \\ Z_2 &= XYg^{\phi} \mod p \\ Z_2 &= XYg^{-\phi} \mod p \end{aligned}$$

4. For $i = 1, 2, 3$, the two talliers run steps 3 to 6 of the single element equality test protocol, with Δ replaced by Z_i. If any of the three runs outputs positive result (meaning the two numbers being tested are equal), output $d_1 = d_2$. Otherwise output $d_1 \neq d_2$.

Theorem 4. *Let O be the output of protocol EQUALITY-TEST-V. Let $O = 1$ if the protocol concludes that $d_1 = d_2$ and 0 otherwise. If $d_1 \neq d_2$, the probability that the protocol (incorrectly) outputs 1 is at most*

$$Pr(O = 1) \leq \frac{1}{\phi}$$

Proof. Let $\delta = d_1 - d_2$. Note that $c \cdot \delta = c \cdot ((u_1 + v_1) - (u_2 + v_2)) \mod \phi = x + y \mod \phi$. Consider the m-dimensional vector space W over $\mathbb{Z}_\phi$. W has ϕ^m elements. For any $\delta \in \mathbb{Z}_\phi^m$, there are at most ϕ^{m-1} vectors in this space that are orthogonal to it, i.e. those on a hyperplane V which is a codimension-1 vector subspace of W and has size ϕ^{m-1}. Since δ is not known to either tallier and the c is randomly drawn, we have

$$Pr(c \cdot \delta = 0) \leq \frac{1}{\phi}$$

Now we show that the protocol actually tests on $c \cdot \delta$. Note that $Z_1 = XY \mod p$ encodes $x + y$. And $c \cdot \delta = x + y \mod \phi = x + y + b$ where b is one of 0 or $\pm\phi$. Clearly $x + y \mod \phi = 0$ is equivalent to one of $x + y, x + y + \phi$ and $x + y - \phi$ is 0, which is what the protocol tests. $\square$

In terms of privacy, note that the talliers compute the projection of the user vectors on a random direction and then test equality on the projections. Due to the zero-knowledge property of the EQUALITY-TEST protocol, this protocol does not leak information either.

In practice, ϕ is typically 2^{32} or 2^{64}, so that the number fits into a machine word. This gives a failure probability of 2.4×10^{-10} or 5.4×10^{-20}. This should be enough for most applications.

The protocols described in Section 2.2.2 and 2.2.2 represent two ends of a spectrum, i.e., the equivalence relation is defined by a single element and the whole vector, respectively. If we want to test only certain elements of the user vectors, we can use the second protocol but with challenge vectors having 0's at certain places to mask out irrelevant elements.

Chapter 3

SVD with Privacy: P4P Style

In the following three chapters we demonstrate how the P4P framework can be used to support private computation of popular algorithms using concrete examples. They include SVD, link analysis and association rule mining. We compare their round complexity and accuracy with their direct, non-private implementations and show that the P4P implementation provides comparable or identical performance.

3.1 Singular Value Decomposition

Recall that for a matrix $A \in \mathbb{R}^{n \times m}$, there exists a factorization of the form

$$A = U \Sigma V^T \tag{3.1}$$

where U and V are $n \times n$ and $m \times m$, respectively, and both have orthonormal columns. Σ is $n \times m$ with nonnegative real numbers on the diagonal sorted in descending order and zeros off the diagonal. Such a factorization is called a singular value decomposition of A. The diagonal entries of Σ are called the the singular values of A. The columns of U and V are left- resp. right-singular vectors for the corresponding singular values.

SVD is a very powerful technique that forms the core of many data mining algorithms and thousands of algorithms in signal processing. Examples include clustering, pattern recognition, Principal Component Analysis (PCA), image and text mining, collaborative filtering, social analysis, etc. Perhaps the most important property of SVD is its ability to optimally approximate the original matrix in a smaller subspace. To see this, let $r = rank(A)$ and u_i, v_i be the column vectors of U and V, respectively. Equation 3.1 can be rewritten as

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

where σ_i is the ith singular value of A.

Let $k \leq r$ be an integer parameter, then we can approximate A by

$$A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{3.2}$$

It is known that of all rank-k approximations, A_k is optimal in Frobenius norm sense, i.e.

$$\|A - A_k\|_F = \min_{rank(B) \leq k} \|A - B\|_F$$

and the k columns of U_k give the optimal k-dimensional approximation to the columnspace of A, while the k rows of V_k^T give the best k-dimensional approximation to the rowspace of A.

Using A_k in place of A projects actual data into a low-dimensional space. For example, in collaborative filtering, there can be hundreds to millions of items (this is the range of m). The dimension k is typically less than ten. On the other hand, this dimension reduction via SVD preserves the structure of original data while considers only essential components of the matrix. This usually filters out noise and uncertainty and improves the performance of data mining tasks. The effectiveness has been shown in works such as Latent Semantic Analysis/Indexing (LSA/LSI) [78], clustering etc.

3.2 ARPACK

Our implementation uses ARPACK [80]. ARPACK (ARnoldi PACKage) is a software package consisting of a collection of Fortran77 subroutines for solving large scale Hermitian, non-Hermitian, standard or generalized eigenvalue problems. The package implements the Implicitly Restarted Arnoldi Method (IRAM) and allows one to compute a few, say k, eigenvalues with user specified features such as those of largest magnitude. Its storage complexity is $nO(k) + O(k^2)$ where n is the size of the matrix. Eigenvectors are also available upon request.

ARPACK is a freely-available yet powerful tool capable of solving large scale eigenvalue problems from significant application areas. Industrial-scale problems with as many as 250,000 degrees of freedom have been solved with this package. It is also generally the fastest and most dependable among the codes that are publicly available [79] and is used internally by MATLAB routines such as `eigs`, `arpackc` and `svds`.

ARPACK is best suited for applications whose matrices are either sparse or not explicitly available: it only requires the user code to perform some "action" on a vector, supplied by the solver, at every IRAM iteration. This action is simply matrix-vector product in our case. ARPACK features a reverse communication interface that frees the user from any particular data structure formats and provides the user with great flexibility and control over how to perform matrix-vector multiplication. As we will show in the next section, we can combine this reverse communication approach with P4P's private vector addition to compute the singular value decomposition of a matrix that is partitioned among the users.

3.3 The Private SVD Scheme

In P4P and with user data, an SVD problem can be formulated as follows. Let $A \in \mathbb{R}^{n\times m}$ be a $n \times m$ matrix of real numbers such that each user i owns a row. We use $A_{i*} \in \mathbb{R}^m$ to denote the m-dimensional row vector owned by user i. In a collaborative filtering application, this vector can be the user ratings. In user activity mining, it can encode information (e.g. frequency) of user's access to documents. In all cases we want to keep this vector private.

Let $k < m$ be a positive integer. The goal is to compute k largest singular values of A and their corresponding singular vectors. In many context such as LSA and collaborative filtering, k is the dimensionality of the reduced linear space that best approximates A in a least square sense.

From equation 3.1, and the fact that both U and V are orthonormal, it is clear that $A^TA = V\Sigma^2V^T$ which implies that

$$A^TAV = V\Sigma^2 \tag{3.3}$$

This means that the singular values of matrix A are just the square roots of the eigenvalues of A^TA and the right-singular vectors of A are the eigenvectors of A^TA. Similarly we have

$$AA^TU = U\Sigma^2 \tag{3.4}$$

and it shows that the eigenvectors of AA^T are the left-singular vectors of A.

Equations 3.3 and 3.4 show the connection between SVD and the symmetric eigenproblem and give us a way to perform SVD using P4P's vector aggregation and ARPACK. Namely the server will host an ARPACK engine and interact with its reverse communication interface. In our case, since both A^TA and AA^T are symmetric, the server program mainly uses `dsaupd`, the ARPACK double precision routine for symmetric problems. Among the arguments passed to `dsaupd`, the server program should ask for k largest (in magnitude) eigenvalues.

At each iteration, `dsaupd` will return a vector $v \in \mathbb{R}^m$ to the server code and ask for the matrix-vector product A^TAv. To see how this is done with P4P's private vector aggregation, notice that

$$\begin{aligned} A^TAv &= [A_{1*}^T, A_{2*}^T, \ldots, A_{n*}^T] \begin{bmatrix} A_{1*} \\ A_{2*} \\ \vdots \\ A_{n*} \end{bmatrix} v \\ &= \sum_{i=1}^{n} A_{i*}^T A_{i*} v \end{aligned}$$

Each term in the summation, $A_{i*}^TA_{i*}v \in \mathbb{R}^m$, is computable by each user using private data. We can treat it as a vector private to each user and input it to the P4P computation which aggregates the vectors across all users without leaking any information about each individual one. The above equation shows that the aggregate is just the matrix-vector product that can be returned to ARPACK. This process is illustrated in figure 3.1.

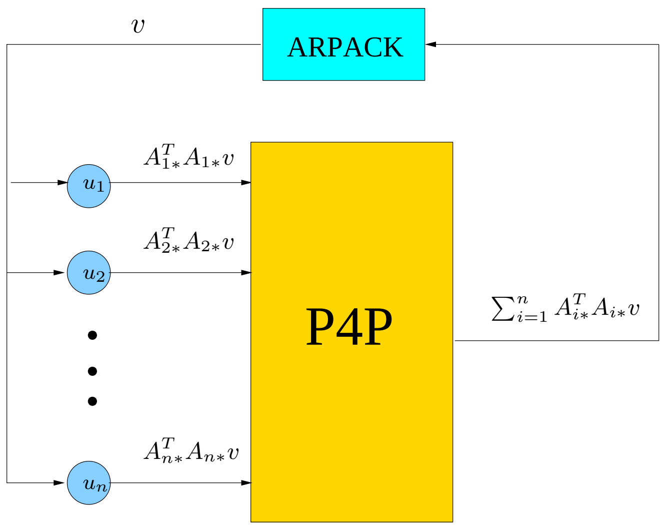

Figure 3.1. Privacy-preserving Singular Value Decomposition (SVD) with P4P.

3.4 Dealing with Real Numbers

In their simplest form, The P4P protocols, like other schemes based on cryptography, only support computation on integers. In most domains, however, applications typically have to handle real numbers. In the case of SVD, even if the original input matrix contains only integer entries, it is likely that real numbers appear in the intermediate (e.g. the vectors returned by ARPACK) and the final results.

Because of the linearity of the P4P computation, we can use a simple scaling, and rounding (to the nearest integers), scheme to convert between real numbers in the application domain and Z_ϕ, P4P's integer field. Let $R > 0$ be the bound of the maximum absolute value application data can take, i.e. all numbers produced by the application are between $[-R, R]$. Suppose $|R| = l$ and $|\phi| = L$. The integer field provides L bits resolution. This means the maximum quantization error for one variable is $R/\phi = 2^{l-L}$. Summing across all n users, the worst case absolute error is bounded by $n2^{l-L}$. In practice L can be 64, and l can be around 20 (this gives a range of $[-2^{20}, 2^{20}]$). This gives a maximum absolute error of 1 over a million.

The scaling factor used must be global, meaning that it should be the same for all users because in the end the scaled vectors will be added together. However, it is difficult to predict before the computation starts and unlikely to stay static during the iterations. This is because, since A is unknown, it is hard to predict the magnitude of the entries in v, the vector returned by ARPACK `dsaupd` routine during one iteration, and $A^T Av$, the resulting vector returned to ARPACK. Ideally one would want to map the number with the largest magnitude that would appear in the computation to the element in Z_ϕ with the largest magnitude and the same sign. This is to save maximum number of bits for precision without causing overflow. [1] Since each $A_{i*}^T A_{i*} v$ is computed by individual users locally, it is not possible to find the largest entry of $A^T Av$ other than using an expensive multiparty computation protocol. To address this problem, we let the server scale the vector v before it sends it to the users so that it is guaranteed that no overflow would happen. The scheme is as follows.

Let a be an estimate of the largest magnitude of $A(i,j)$, i.e. $a \geq |A(i,j)|$ for $i = 1, \ldots, n, j = 1, \ldots, m$. The server finds $b = \max_i |v[i]|$ and computes

$$v' = \alpha v \tag{3.5}$$

where $\alpha = mna^2 b/R$. v' is then sent to all users. Each user i will convert v' to integers in Z_ϕ by scaling it by a factor of $\phi/(2R)$ and rounding to the nearest integers. [2] We assume that there is a procedure `rtoi` that performs such conversion. Similarly we also assume there is another procedure, `itor`, that converts the integers in Z_ϕ back to real numbers in $[-R, R]$ by scaling them by $2R/\phi$. Let $d_i = \texttt{rtoi}(v')$. It is straightforward to verify that entries in $\sum_{i=1}^n d_i$ are all within Z_ϕ, i.e. no overflow happens.

The server then simply converts the aggregate output by P4P back to real numbers using `itor` and scales it by $1/\alpha$. This will be the vector returned to the ARPACK routine.

3.5 The Protocol

The protocol is carried out as follows:

[1]Note that we must keep all integers, including the sums, within Z_ϕ, otherwise P4P gives incorrect aggregate.

[2]It is easy to see that this scaling maps R to $\lfloor \phi/2 \rfloor$.

1. Whenever the ARPACK routine returns control to the server program with a vector $v \in \mathbb{R}^m$, the server computes v' using equation 3.5. It then broadcasts v' to all users.

2. If the number of iterations so far exceeds γm^2, where $\gamma < 1$ is a parameter, the user terminates the protocol. Otherwise user i computes

$$w_i = A_{i*}^T A_{i*} v'$$

   and then converts w_i into integers using the scheme described earlier and obtains $d_i = \texttt{rtoi}(w_i)$.

3. User i inputs d_i into the P4P protocol.

4. After the P4P protocol outputs $d = \sum_{i=1}^{n} d_i$, the server converts it to real domain:

$$w = \texttt{itor}(d)$$

   and returns w/α as the matrix-vector product to `dsaupd` which runs another iteration.

5. When the ARPACK routine indicates convergence or stops after certain number of iterations, the server outputs

$$\begin{aligned} \Sigma_k &= diag(\sigma_1, \sigma_2, \ldots, \sigma_k) \in \mathbb{R}^{k \times k} \\ V_k &= [v_1, v_2, \ldots, v_k,] \in \mathbb{R}^{m \times k} \end{aligned} \tag{3.6}$$

   with $\sigma_i = \sqrt{\lambda_i}$ where λ_i is the ith eigenvalue and v_i the corresponding eigenvector computed by ARPACK, $i = 1, \ldots, k$, and $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_k$.

3.6 Privacy Analysis

Note that the protocol does not compute U_k. This is intentional. U_k contains information about user data: the i^{th} row of U_k encodes user i's data in the k-dimensional subspace and should not be revealed at all in a privacy-respecting application. We note that in many applications most of the desired information can be computed from the singular values (Σ_k) and the right singular vectors (V_k^T) which gives the best k-dimensional approximation to

Table 3.1. Datasets

Dataset	Dimensions	Density	Range	Type
Enron	150×150	0.0736	$[0, 1593]$	Social graph
EM	74424×1648	0.0229	$[0, 1.0]$	Movie ratings
RAND	2000×2000	1.0	$[-2^{20}, 2^{20}]$	Random

the rowspace of A and is a good representation of *anonymous* user data. For example, in collaborative filtering where A_{ij} encodes user i's rating on item j, user i can generate recommendations for herself using Σ, V^T, and A_{i*} which is local and private to the user [21].

At each iteration, the protocol also reveals the matrix-vector product $A^T Av$. This should not be much of a privacy concern because the final results Σ_k and V_k^T already give an approximation of $A^T A$ (note that $A^T A = V\Sigma^2 V^T$) therefore the intermediate aggregates do not reveal more information. In addition, $A^T A$ has m^2 entries and the user will abort the protocol if the number of iterations exceeds γm^2. This prevents the server from collecting enough information to obtain a more accurate estimate of $A^T A$ from the products. The parameter γ is used to control how conservative the system is in protecting privacy. From our experiments, γ typically can be 0.01 or smaller and still allows for enough iterations for ARPACK to converge.

3.7 Implementation and Evaluation

We implemented the SVD protocol together with the P4P framework and performed some tests. For the work reported in this chapter, we have created a Java wrapper of ARPACK (which is in Fortran77) to allow it to be integrated with our P4P code. The Java wrapper makes ARPACK much easier to use and may be of independent interests. It will also be released together with the P4P code.

We measured two quantities in all our tests: N, the number of IRAM iterations until ARPACK indicates convergence, and ϵ, the relative error. The former is the number of matrix-vector computation that was required for the ARPACK to converge. It is also the number of times P4P aggregation is invoked. Note that this number is determined by a number of factors including the nature of the problem, the IRAM convergence rate and possible errors introduced by converting between real and integer numbers for P4P aggregation. To see how our privacy mechanism affects N, we also measured it using ARPACK with a centralized, direct, non-privacy-preserving matrix-vector multiplication. In all our experiments, we found no difference in N between this centralized method and our P4P-based implementation.

Note that N is sensitive to how many Lanczos vectors are generated at each iteration, which is controlled by `ncv`, the 8^{th} argument to the ARPACK `dsaupd` routine. In our tests we attempted to minimize it by trying different `ncv`. But, since our search is not exhaustive, it is possible that our results are not optimal in terms of IRAM's convergence rate. However, this issue also exists in a server-based, non-private solution and is orthogonal to the privacy techniques we developed. What is important is, as our tests indicated, our privacy protection techniques do not affect it in a negative way.

The error ϵ is computed as

$$\epsilon = \max_{i=1,\ldots,k} |\frac{A^T A v_i - \lambda_i v_i}{\lambda_i}|$$

i.e. it measures the maximum relative residual norm among all eigenpairs computed.

We evaluated our implementation on three datasets: the Enron Email Dataset [31], EachMovie (EM), and a randomly generated dense matrix (RAND). The Enron data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. The corpus contains email data from 150 users, spanning a period of about 5 years (Jan. 1998 to Dec 2002). Our test was run on the social graph defined by the email communications. The graph is represented as a 150×150 matrix A with $A(i, j)$ being the number of emails sent by user i to user j. EachMovie dataset is a well-known test dataset for collaborative filtering algorithms [17]. This dataset comprises ratings of 1648 movies by 74424 users. Each rating is a number in the range $[0, 1]$. Both the Enron and EachMovie datasets are very sparse, with densities of 0.0736 and 0.0229, respectively. To test the performance of our protocol on dense matrices, we generated ran-

Table 3.2. Enron Social Graph

k	10	20	30	40	50	60	70	80	90	100
N	67	97	122	162	109	137	172	167	171	169
$\epsilon(\times 10^{-8})$	0.00049	0.0021	0.0046	0.0084	0.0158	0.0452	0.121	0.266	0.520	1.232

domly a 2000×2000 matrix with entries chosen in the range $[-2^{20}, 2^{20}]$. The properties of the datasets are summarized in table 3.1.

In all the experiments, ϕ is set to be a 62-bit integer and the P4P key is 1024-bit.

Table 3.2 to 3.4 summarize the experiments results for the thress datasets. In all these tests, we used machine precision as the tolerance input to ARPACK. The accuracy we obtained is very good. The relative residual norm remains very small for all tests (10^{-12} to 10^{-8}). This should be far more enough for most applications.

In terms of round complexity, N ranges from under 100 to a few hundreds. As we mentioned earlier, this is an orthogonal issue to our privacy technique and we did not optimize for it in our experiments. We stress that, for comparison, we also measured the number of iterations required by ARPACK when we perform the matrix-vector multiplication directly without the P4P aggregation. In all our experiments, we found no difference in N between this direct method and our privacy preserving implementation.

In our P4P implementation, with 1024-bit key length, it takes 74.73 seconds of server time to validate and aggregate all 150 Enron user data on a *single* machine (each user needs to spend 726 milliseconds to prepare the zero-knowledge proofs). This translates to a total of 5000 seconds or 83 minutes spent on private P4P aggregation to compute $k = 10$ singular-pairs. To compute the same number of singular pairs, which is typical of collaborative filtering applications, for EachMovie, aggregating all users data takes about 6 hours (again on a single machine) and the total time for 70 rounds is 420 hours. This total seems large. But note that the data miner's aggregation process is trivially parallelizable. It is unlikely a data miner would service 74424 user with a single PC. Using a cluster of, say 50 nodes, will reduce the running time to about 10 hours. It should be feasible to run the protocol over several days as a background process. Since the user ratings data are changing slowly, a few days latency does not diminish the value of the aggregate.

Table 3.3. EachMovie

k	10	20	30	40	50	60	70	80	90	100
N	70	140	254	222	276	371	322	356	434	508
$\epsilon(\times 10^{-12})$	0.470	0.902	1.160	1.272	1.526	1.649	1.687	2.027	2.124	2.254

Table 3.4. RAND

k	10	20	30	40	50	60	70	80	90	100
N	304	404	450	480	550	700	770	720	810	800
$\epsilon(\times 10^{-9})$	3.996	3.996	3.996	3.996	3.996	3.996	3.996	3.996	3.996	3.996

Table 3.5. EachMovie (with 10^{-3} stopping criterion)

k	10	20	30	40	50	60	70	80	90	100
N	25	50	90	120	150	150	175	200	225	250
$\epsilon(\times 10^{-4})$	2.873	4.982	0.0989	0.0016	0.000004	4.099	0.6506	0.2267	0.02708	0.00431

N can be significantly reduced if the application can tolerate a higher error. This is achieved by setting the stopping criterion of ARPACK's `dsaupd` routine to a small positive value. For example, using a stopping criterion of 10^{-3} (we were using machine precision for previous experiments), the number of iterations can be reduced by as much as more than half. And the residual norm ϵ should be still acceptable (below 10^{-3}) for most applications. [3] Table 3.5 shows the results for EachMovie dataset. Similar results can be observed with other two datasets as well.

[3]Note that the stopping criterion is an input to `dsaupd`, not the residual norm we are measuring.

Chapter 4

Online Link Analysis on Dynamic Weighted Graph

4.1 Introduction

Link analysis algorithms have been used successfully on hyperlinked data to identify authoritative documents and retrieve other information. For instance, the expertise location problem [110, 73, 74, 86] is to find a person in an organization or community who is knowledgeable (and authoritative) in an area. Several approaches [110, 73, 74] construct an explicit social network between individuals, based on email or similar logs, and then use graphical analysis to locate the relevant experts. Similarly, the document ranking problem is to determine the relative levels of "authoritativeness" among a collection of documents. Link analysis algorithms have been used in these environments to uncover such information [77, 18].

The primary drawback to the above approaches is the need for explicit structure about the social relations between individuals and the hyperlinks among documents, which do not necessarily exist. For instance, in a computer-mediated environment, a group of individuals could be using tools like software applications to access documents collaboratively, and there is neither an explicit social network representing how each individual is related to others, nor hyperlinks among documents. However, in such context, there are still compelling needs in identifying domain experts and authoritative documents.

Another inadequacy of such algorithms, when applied to authoritative document ranking, as Kleinberg acknowledged [77], is that they only make use of the structural information about the graph as defined by the links, and fail to capture patterns of user access which encode essential information about the user's attitude toward the document. The intuition behind the link analysis algorithms is that the link structure encodes important information about the nodes. For example, according to [77], the links among documents, be it hyperlinks on www or citations among academic papers, are constructed consciously by the authors of the documents and represent the authors' "endorsement" toward the authority of documents pointed to by the links and the HITS algorithm [77] can uncover such information to produce a ranking of documents according their level of authority. We believe that a similar principle also holds with patterns of user access: the way a user accesses a document could reflect his/her opinion about it. Meanwhile, a user's level of expertise can also be reflected by the documents that he/she accesses. There is a mutually reinforcing relationship between these two measures, which maps naturally to what Kleinberg denotes "hub" and "authority" [77]: a person is more likely to be an expert in an area if he/she reads more authoritative documents and a document is more likely to be authoritative if it is read by many experts. This phenomenon can also be observed in other graphs such as social networks where the structure is implicitly defined by communication.

In this chapter we propose an approach to address these limitations and describe an algorithm that is suitable for such purpose. Notice that access pattern and link structure are not mutually exclusive. Rather, access pattern can complement or even define the other. Our approach uses weighted graphs to model the relationship among nodes and the weights can encode user access or communication. In situations where no explicit link structure exists, these weights effectively *define* the graphical structure and link analysis algorithms can be applied. Where there is an explicit link structure, weights obtained from access or communication analysis can be used to *augment* existing graph and uncover more information.

Using weights in identifying authoritative documents is not new [108]. The novelty of this work is that we propose the use of weights to model user behavior and construct the link structure. This enables us to apply link analysis algorithms in settings where no such structure exists. However, computing on patterns of access or communication has two implications: (1) instead of a static system, the graph becomes dynamic. The model changes as more data is observed; and (2) user privacy becomes an issue due to the sensitive nature of the user's information used to construct and update the graph. (1) may also mandate that the system services users' query in real-time as there is no end to the accumulation of observations.

Meanwhile, emergence of new applications and changes of existing ones are making these problems more acute. For example, search engines are moving towards personalization (e.g. http://labs.google.com/personalized) that will be relying heavily on client side logging. While link analysis remains the core of their ranking algorithm (e.g. Google's PageRank [18]), the input data will be augmented by user configuration and access monitoring. Other applications in fields such as knowledge management, IR, HCI and data mining also showed similar trend.

To address these new issues, we devised Secure OnlineHITS, a distributed version and enhancement of Kleinberg HITS algorithm that (1) amortizes cost across a number of updates by using "lazy updates", which makes it more suitable for dynamic environments; and (2) uses cryptographic techniques to preserve user's privacy while performing the computation. Our technique can be easily extended to other link analysis algorithms such as Google's PageRank [18] but we won't pursue it in this chapter. To make it concrete, we describe the algorithm in the context of document and expertise ranking. However, it is general enough to be applied to other situations where link analysis is appropriate. We use the term document in a broad sense. It refers to any information that can be identified, accessed and analyzed as a unit. For example, a web page or an image can all be classified as a document.

In the rest of the chapter, we first review the original HITS algorithm in Section 4.2. We then discuss the construction of a weighted graph and prove the convergence of a HITS-like algorithm on such graphes in Section 4.3. In Section 4.5 we derive an online version of the HITS algorithm to make it more efficient to run in a dynamic environment on accumulated data. Evaluations are presented in Section 4.6. Finally in Section 4.7 we discuss privacy and security issues in running such kind of user activity analysis and describe our privacy preserving implementation of online HITS based on public-key encryption.

4.2 A Review of HITS

Kleinberg's HITS algorithm [77] is a well-known link analysis algorithm that identifies "authoritative" or "influential" webpages in a hyperlinked environment. Intuitively, by thinking of a hyperlink as a citation, a webpage i is more of an authority (i.e. highly-referenced page) as compared to webpage j if there are more hyperlinks entering i from hub webpages, where a hub is simply a webpage that is a valuable source of links to other webpages. Likewise, a webpage i is a better hub than webpage j if there are more hyperlinks exiting i into authoritative webpages. Given a set of n webpages, HITS first constructs the corresponding n-by-n adjacency matrix A, such that the element in row i and column j of A is 1 if there exists a hyperlink from webpage i to webpage j, 0 otherwise. HITS then iterates the following equations:

$$x^{(t+1)} = A^T y^{(t)} = (A^T A) x^{(t)} \tag{4.1}$$

$$y^{(t+1)} = A x^{(t+1)} = (A A^T) y^{(t)} \tag{4.2}$$

Where the i-th element of x denotes the authoritativeness of webpage i and the i-th element of y denotes the value of webpage i as a hub. With the vectors x and y initialized as vectors of ones and renormalized to unit length at every iteration, as t approaches infinity, $x^{(t+1)}$ and $y^{(t+1)}$ approach x^* and y^*, the principal eigenvectors of $A^T A$ and AA^T, respectively.

Even though HITS is originally intended to locate hubs and authorities in a hyperlinked environment, we observe that hubs and authorities map very well to the users and documents in access based link analysis and the relationship of mutual reinforcement still holds as mentioned in Section 4.1.

4.3 Constructing a Weighted Graph

By observing users behavior we can construct a graph of users/documents in environments where no such structure exists. We assume we can observe users' document access and communication pattern using tools like client side logger. Such tools are available from a number of sources. In particular, we have developed a version of our own that that can monitor user's document access and email communication. Of course such tools have serious privacy implications and we will address such issue in Section 4.7.

The system consciously logs the users' activities as tuples of the form $< i, j >$, which denotes the fact that user i accesses document j or communicates with user j, depending on the context. These log entries represent tacit data about the collaborative context because they do not directly encode the links between users nor documents. Given this activity log, we can construct a graph, such that vertices represent the users and/or documents and an edge (i, j) exists and has non-negative weight $w_{i,j}$ iff an item $< i, j >$ exists in the activity log.

How the weight $w_{i,j}$ is computed depends on the application and the goal of the link analysis. The ideas such as TFIDF [108], and the power law of practice [94], etc, are all good heuristics. In some situations the weight can be reduced to simple access or message count. This decision is orthogonal to our work and won't be pursued in this chapter. The only assumption we make here is that $w_{i,j}$ is a non-negative, real number.

4.4 Convergence of Weighted HITS

Suppose we replace the 0-1 valued element A_{ij} in the adjacency matrix A with a non-negative weight function $w(i, j)$. First we introduce the following two lemmas from [95].

Lemma 2. *If M is a symmetric matrix and v is a vector not orthogonal to the principal eigenvector v^* of M, then the unit vector in the direction of $M^k v$ converges to v^* as k increases without bound.*

Lemma 3. *If a symmetric M matrix has only non-negative elements, the principal eigenvector of M has only non-negative entries.*

According to the definition of $w(i, j)$, it's easy to see that matrix A has only non-negative values and the symmetric matrix A^TA and AA^T have only non-negative values, thus the principal eigenvectors of A^TA and AA^T have only non-negative entries (lemma 3).

In running HITS on a weighted graph, if we start with a vector x with non-negative entries, since x is not orthogonal to the eigenvector of AA^T which has only non-negative entries, the sequence $\{y^{(k)}\}$ converges to a limit y^* (lemma 2). Similarly we can prove that the sequence $\{x^{(k)}\}$ converges to a limit x^*.

4.5 Online HITS

Access based graph construction and link analysis introduces a number of issues of its own such as frequent update, distributed data sources, data security and user privacy concerns, etc. An algorithm alone cannot address all these issues. But a properly designed algorithm can make addressing them a lot easier. In this section we describe a link analysis algorithm that works incrementally as data is being added. We use the idea of "lazy update" to avoid updating and running of the expensive computation so that we can amortize the cost across a number of updates while still maintaining enough precision.

4.5.1 Basic Approach

As shown in Section 4.1 and 4.3, the intuition behind HITS fits very well to our application. However, the algorithm is too expensive to run on every update, which can be quite frequent. Recall that the rankings we are seeking, x and y, correspond to the principal eigenvectors of A^TA and AA^T, respectively. A key observation is that a single update to the user access traffic corresponds to a perturbation to the A matrix. Depending on the weight function selected, it can perturb a single element or a row of A. In either case the perturbation is local. This perturbation will cause variation to the principal eigenvector of

A^TA (and AA^T). If we can find the relationship between the variation of x and y and the perturbation to A, we can check each update to see if it will cause too much variation to x and y. If the change is within acceptable tolerance, we can postpone applying the update thus avoiding running HITS for it. When the accumulated updates cause too much perturbation, we apply them together and run HITS once. This is essentially an approximation to HITS that amortizes its cost across multiple updates. We denote such an algorithm Online HITS. Another advantage of this approach is that service of user queries and updating A and running of HITS can be made separate. The system can update A and run HITS in background and continue servicing user queries with old results that we are confident to be within certain range from the latest ones. Users can enjoy nonblocking service.

Similar issues have been discussed in the context of stability of the HITS algorithm [96, 97]. However, there is a subtle but significant difference between our approach and theirs: we are not concerned with the incompleteness of our data or the stability of the results. For us, the everlasting accumulation of data is an inherent feature of our system and the results we produce are the "best guess" based on the data we have so far. It is perfectly alright for the results to undergo dramatic change, which reflects the update of the system's knowledge about the world. Rather, we are interested in the bound of the change so that we can perform the tasks more efficiently. In addition, the conclusions in [96, 97] only apply to unweighted graphs represented by adjacency matrices. The theorem we describe below is applicable to any weighted graph.

Online HITS is based on the following theorem:

Theorem 5. *Let $S = A^TA$ be a symmetric matrix. Let a^* be the principal eigenvector and δ the eigengap[1] of S. Let E_S be a symmetric perturbation to S. We use $\|\cdot\|_F$ to denote the Frobenius norm[2]. For any $\epsilon > 0$, if $\|E_S\|_F \leq min\ \{\frac{\epsilon\delta}{4+\sqrt{2}\epsilon}, \frac{\delta}{2\sqrt{2}}\}$, then the principal eigenvector $\tilde{a}^*$ of the perturbed matrix $\tilde{S}$ satisfies*

$$\|a^* - \tilde{a}^*\|_2 \leq \epsilon$$

[1] *Eigengap is defined to be the difference between the largest and the second largest eigenvalues.*

[2] *The Frobenius norm of a matrix X is defined by $\|X\|_F = (\sum_i \sum_j (X_{ij})^2)^{1/2}$*

Proof. We use $\tilde{.}$ to represent perturbed quantity. Suppose $S \in \mathbf{R^{n\times n}}$ is a symmetric matrix with principal eigenpair (λ^*, a^*), and eigengap $\delta > 0$. Let E_S be a symmetric perturbation to S such that $\tilde{S} = S + E_S$. By Theorem V.2.8 from matrix perturbation theory[118], there is *some* eigenpair of $\tilde{S}$ $(\tilde{\lambda}, \tilde{a})$ such that

$$\|a^* - \tilde{a}\|_F \leq \frac{4\|E_S\|_F}{\delta - \sqrt{2}\|E_S\|_F} \tag{4.3}$$

and

$$|\lambda^* - \tilde{\lambda}| \leq \sqrt{2}\|E_S\|_F \tag{4.4}$$

assuming the denominator in 4.3 is positive. Let $L \in \mathbf{R^{n-1\times n-1}}$ be diagonal matrix containing all S's eigenvalues except λ^*. A bound similar to 4.4 holds:

$$\|L - \tilde{L}\|_F \leq \sqrt{2}\|E_S\|_F \tag{4.5}$$

Let $\tilde{\lambda_2}$ be the largest eigenvalue in $\tilde{L}$. By Corollary IV.3.6 of [118], Equation 4.5 implies

$$\tilde{\lambda_2} \leq \lambda_2 + \sqrt{2}\|E_S\|_F \tag{4.6}$$

Since $\|E_S\|_F \leq \frac{\delta}{2\sqrt{2}}$, Equations 4.4 and 4.6 ensures that $\tilde{\lambda} > \tilde{\lambda_2}$, i.e. $(\tilde{\lambda}, \tilde{a})$ is indeed the principal eigenpair of $\tilde{S}$. Also this will ensure the denominator in 4.3 is indeed positive.

Given any $\epsilon > 0$, if $\|E_S\|_F \leq \frac{\epsilon\delta}{4+\sqrt{2}\epsilon}$, then $\frac{4\|E_S\|_F}{\delta-\sqrt{2}\|E_S\|_F} \leq \epsilon$ thus we have $\|a^* - \tilde{a}\|_2 \leq \epsilon$. $\square$

This theorem gives us a way to test the perturbation and bound the error on the principal eigenvector. The proof is similar to that presented in [97] and is given in appendix.

There are two subtle issues that need to be addressed before we can use this theorem to construct an online HITS algorithm, namely the computations of eigengap δ and perturbation $\|E_S\|_F$. They have to be performed efficiently otherwise the cost of computing them would offset the saving of not running HITS. They will be addressed in the following subsections.

4.5.2 Computation of Eigengap

A straightforward way of computing eigengap δ is to calculate λ_1 and λ_2, the largest and the second largest eigenvalues, and take the difference. The original HITS algorithm is essentially a power method to compute the principal eigenvector of S. It can be revised easily, without adding complexity, to produce λ_1 and λ_2 as byproducts. Two modifications to the original HITS algorithm are introduced:

1. Instead of finding only the principal eigenvector, find the two eigenvectors corresponding to λ_1 and λ_2. This can be done by using the "block power method" ([119], pp. 289). Concretely, start with two orthogonal vectors, multiply them all by S, then apply Gram-Schmidt to orthogonalize them. This is a single step. Iterate until they converge.

2. HITS normalizes the vector at each step to unit length. This is not necessarily the only choice to ensure convergence. Instead, we normalize each vector by dividing them by their first non-zero element. They still converge to the two eigenvectors and the scaling factors converge to λ_1 and λ_2 ([119], pp. 289).

The above modifications introduce extra computation of one eigenvector and Gram-Schmidt orthogonalization. The former doubles the work of HITS and the latter is $O(n)$. The total complexity is the same as HITS: $O(mn)$.

4.5.3 Upper Bound of $\|E_S\|_F$

Let E be perturbation to matrix A (This is our update to the graph). Then $\tilde{A} = A + E$ and $\tilde{S} = (A+E)^T(A+E) = A^TA + A^TE + E^TA + E^TE$. Let $E_S = A^TE + E^TA + E^TE$. We know for Frobenius norm (actually for any norms) $\|X + Y\|_F \leq \|X\|_F + \|Y\|_F$. So $\|E_S\|_F \leq 2\|A^TE\|_F + \|E^TE\|_F$. This bound involves matrix multiplication which we try to avoid. Note that the purpose of our online HITS is to postpone running the algorithm so that we can save some computation. This means that we will accumulate a number of updates (since the last time we update A and run HITS). Even though each single update is local and involve only one element or one row of A, all the accumulated updates will affect a number of A's elements. This means E can be sparse but unlikely to have only single non-zero element or a row. Let $E(t)$ be the accumulated unapplied update matrix after we observed tth update (we reset the counting each time we apply updates). $E(t) = E(t-1) + \Delta(t)$ where $\Delta(t)$ has only one non-zero element or row. We have

$$\|E_S(t)\|_F \leq 2\|A^T E(t)\|_F + \|E(t)^T E(t)\|_F \tag{4.7}$$

where

$$\begin{aligned} \|A^T E(t)\|_F &= \|A^T (E(t-1) + \Delta(t))\|_F \\ &\leq \|A^T E(t-1)\|_F + \|A^T \Delta(t)\|_F \end{aligned} \tag{4.8}$$

and

$$\begin{aligned} \|E(t)^T E(t)\|_F &= \|(E(t-1) + \Delta(t))^T (E(t-1) + \Delta(t))\|_F \\ &= \|E(t-1)^T E(t-1) + E(t-1)^T \Delta(t) \\ &\quad +\Delta(t)^T E(t-1) + \Delta(t)^T \Delta(t)\|_F \\ &\leq \|E(t-1)^T E(t-1)\|_F \\ &\quad +2\|E(t-1)^T \Delta(t)\|_F + \|\Delta(t)^T \Delta(t)\|_F \end{aligned} \tag{4.9}$$

The three equations above give us a way to compute the upper bound on $\|E_S\|_F$ recursively. Namely we can keep running updates on the upper bounds of $\|A^T E(t-1)\|_F$ and $\|E(t-1)^T E(t-1)\|_F$ using Equation 4.8 and 4.9, respectively, and add to them the other terms in the equations to get new upper bounds for the next step.

When a single update involves only one element of A, $\Delta(t)$ has a single non-zero element. Let $\Delta_{ij}(t)$ be the non-zero element of $\Delta(t)$, then

$$\begin{aligned} &\|A^T \Delta(t)\|_F = \Delta_{ij}(t)\|A_{i*}\|_2 \\ &\|E(t-1)^T \Delta(t)\|_F = \Delta_{ij}(t)\|E(t-1)_{i*}\|_2 \end{aligned} \tag{4.10}$$

where A_{i*} and $E(t-1)_{i*}$ are the ith row of A and $E(t-1)$, respectively.

There are two ways to compute upper bounds of $\|A^T \Delta(t)\|_F$ and $\|E(t-1)^T \Delta(t)\|_F$: (1) keep the matrix $E(t-1)$ and use Equation 4.10; (2) use the largest elements of A and $E(t-1)$ to estimate. (1) is accurate and involves $O(n)$ operations. (2) is fast (only scalar multiplication). The actual choice depends on application.

When an update changes a row of A, computing $\|A^T\Delta(t)\|_F$ and $\|E(t-1)^T\Delta(t)\|_F$ is more expensive and requires $O(n^2)$ operations and $\|\Delta(t)^T\Delta(t)\|_F = \|\Delta_{i*}(t)\|_2^2$ which is $O(n)$. This is at the same level of complexity as HITS but can be substantially cheaper to run because the latter takes a number of iterations to converge while the former needs to run only once. Kleinberg reported that the typical number of iterations for HITS to converge is 20 [77]. If the cost is still too high to accept, there are two ways to alleviate: (1) Frobenius norm has the property $\|AB\|_F \leq \|A\|_F\|B\|_F$. $\|A^T\Delta(t)\|_F$ and $\|E(t-1)^T\Delta(t)\|_F$ can be reduced to scaler multiplication (with loss of "tightness"); (2) the computation can be made to be distributed across all clients, as described in Section 4.7.

4.5.4 The Algorithm

Putting all these together, we summarize the Online HITS algorithm in this section. In the following, we assume there is a procedure Gram-Schmidt that, given a matrix M, orthogonalizes its column vectors using Gram-Schmidt process ([119], pp. 129). We also assume there is a process that monitors the data and invokes our algorithm with perturbation when it sees an update.

Let $z_n = (1, 1, \ldots, 1)^T \in R^n$. Let $z_n^{\perp} \in R^n$ be the vector that is orthogonal to z_n and has the same length. $\Delta \in R^{n\times m}$ is the perturbation caused by a single update. ϵ is the required precision. Let $x[1]$ be the first non-zero element of vector x. We keep global variables $\|E_S\|_F, \|A^TE\|_F$ and $\|E^TE\|_F$. To make it concise, we use matrix computations in the pseudocode. However, it is clear that they can either be implemented together with HITS iterations, or only require operations on small number of the elements of the matrices involved, as described in Section 4.5.3.

The two main procedures are NewHITS and OnlineHITS. NewHITS is the modified version of HITS algorithm that performs block power iterations on two vectors and compute eigengap. Note that A^TA and AA^T share the none-zero eigenvalues so only one eigengap is needed. OnlineHITS is called on each update. It checks whether all the accumulated updates would cause large perturbation to the ranking. If so it will apply the updates and invoke NewHITS. Otherwise it returns the ranking from previous round. These two procedures are listed below.

```
NewHITS($A$, $\epsilon$)
  $A \in R^{n \times m}$
  $x \leftarrow z_m$, $x_\perp \leftarrow z_m^\perp$
  $y \leftarrow z_n$, $y_\perp \leftarrow z_n^\perp$
  Do
      $x \leftarrow Ay$, $x_\perp \leftarrow Ay_\perp$
      $y \leftarrow A^T x$, $y_\perp \leftarrow A^T x_\perp$
      $[x, x_\perp] \leftarrow$ Gram-Schmidt($[x, x_\perp]$)
      $[y, y_\perp] \leftarrow$ Gram-Schmidt($[y, y_\perp]$)
      $\delta \leftarrow x[1] - x_\perp[1]$
      $x \leftarrow x/x[1], x_\perp \leftarrow x_\perp/x_\perp[1]$
      $y \leftarrow y/y[1], y_\perp \leftarrow y_\perp/y_\perp[1]$
  Until error $< \epsilon$
  Return $(x, y, \delta)$
```

```
OnlineHITS($\Delta$, $\epsilon$)
  $\|A^T E\|_F \leftarrow \|A^T E\|_F + \|A^T \Delta\|_F$
  $\|E^T E\|_F \leftarrow \|E^T E\|_F + 2\|E^T \Delta\|_F + \|\Delta^T \Delta\|_F$
  $\|E_S\|_F \leftarrow 2\|A^T E\|_F + \|E^T E\|_F$
  $E \leftarrow E + \Delta$
  If $\|E_S\|_F > Tol$
      $A \leftarrow A + E$
      $[x, y, \delta]$ = NewHITS($A$, $\epsilon$)
      $E \leftarrow 0$
      $\|A^T E\|_F \leftarrow 0$
      $\|E^T E\|_F \leftarrow 0$
      $\|E_S\|_F \leftarrow 0$
```

```
        $Tol = \frac{\epsilon\delta}{4+\sqrt{2}\epsilon}$
    Endif
    Return $(x, y)$
```

4.6 Evaluation

Compared to HITS, OnlineHITS is at the same complexity level. However, its advantage lies in the hope that the updates may not cause too much perturbation to the ranking so that recomputation is avoided. In addition, the operations introduced for perturbation checking do not require iteration so they are substantially cheaper than HITS. The benefit gained by Online HITS depends on the stability of the system in face of perturbation, which is application-specific. We believe that in situations where data is accumulating, Online HITS is most likely advantageous. The intuition behind this belief is that the more data is accumulated, the less significant a new update would be to the overall ranking. Therefore there would be more opportunities to avoid update and running of HITS.

To evaluate how well Online HITS performs, we implemented the algorithm and ran it on the Enron Email Dataset [31]. We used some of the useful mappings created by Andres Corrada-Emmanuel [32]. In particular, for each email, we used the mappings to find its author and recipients. As pointed out in [32], The Enron corpus contains some inconsistencies. In our test, we ignored emails that were mapped to multiple authors. Multiple recipients of a single email, however, are preserved.

This test can be thought of as “identifying the central figures” in the social network defined by the email communications. In constructing the graph, we simply used message count as the weight for each link between two users. Since one email may have multiple recipients, multiple links may be updated when an email is observed. There are a total of 150 users in the data set and our algorithm ranks them in “importance” according to their email communication.

An email is treated as a log item. Online HITS constantly monitors the log and performs

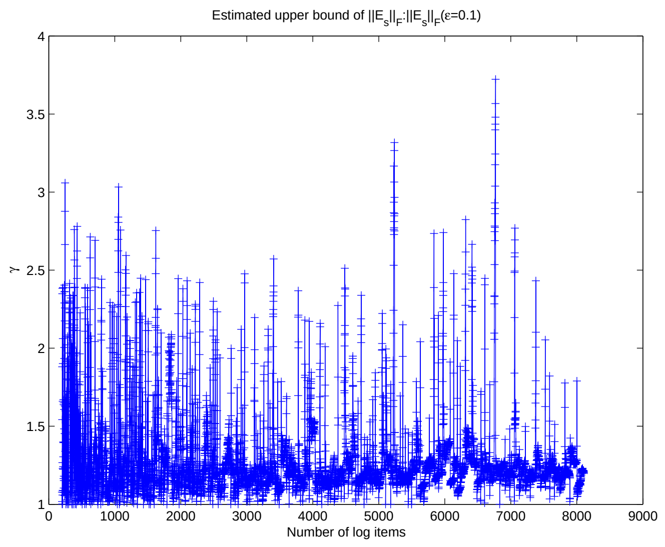

Figure 4.1. Approximation ratio: the ratio of the estimated upper bound of $\|E_S\|_F$ and its actual value.

operations as described in Section 4.5. A total of 8107 log items are observed. The precision is chosen to be $\epsilon = 0.1$ [3].

Note that we are not testing how well the ranking produced by Online HITS (or HITS) fits the "real" ranking which is a rather qualitative and subjective measure. Instead, we are examining Online HITS's algorithmic properties and how it performs more efficient than original HITS in a dynamic system.

The results of our test are shown in the following figures.

Figure 4.1 plots the ratio of the estimated upper bound of $\|E_S\|_F$ and its actual value. I.e. for each update $\gamma = (\|2A^TE\|_F + \|E^TE\|_F)/\|E_S\|_F$. It shows how tight the upper bound given in Section 4.5.3 is. The number varies as updates accumulate and are applied, but never exceeds 3.8.

Figure 4.2(a) shows, for each update, the actual perturbation $\|E_S\|_F$, the upper bound we estimated based on the method of Section 4.5.3, and the tolerance as specified by Theorem 5. Although the details are not easily discernable due to the large number of data points, it clearly shows the general trend of these measures, i.e. the tolerance grows as the

[3]The choice of the precision depends on the application and the data. As we will observe later, the rankings of individual users in the Enron Email Dataset are quite "far" from each other and a larger ϵ can be used without affecting their relative standings. The result will be more saving in computation.

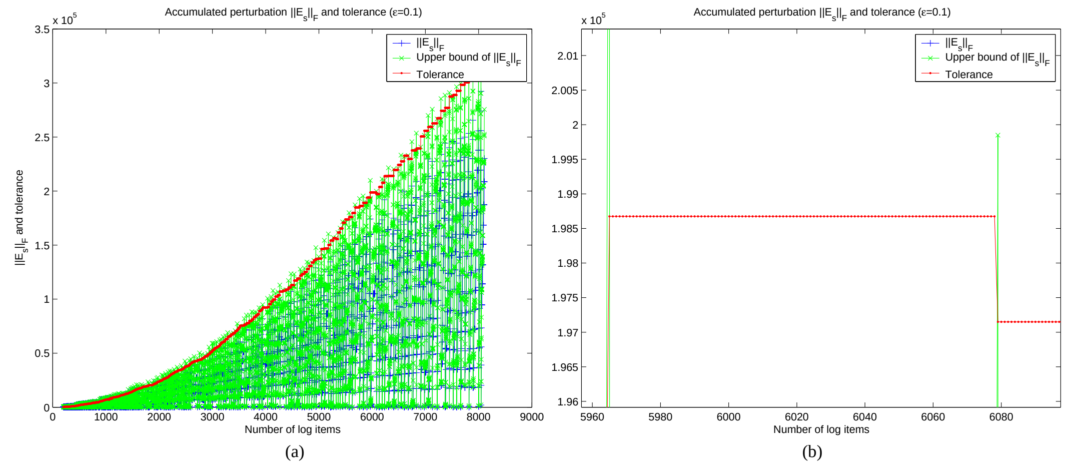

Figure 4.2. Accumulated perturbation $\|E_S\|_F$ and tolerance. (a) shows the actual perturbation $\|E_S\|_F$, the upper bound we estimated based on the method of Section 4.5.3, and the tolerance as specified by Theorem 5. (b) enlarges the area of (a) between data item 5965 to 6078.

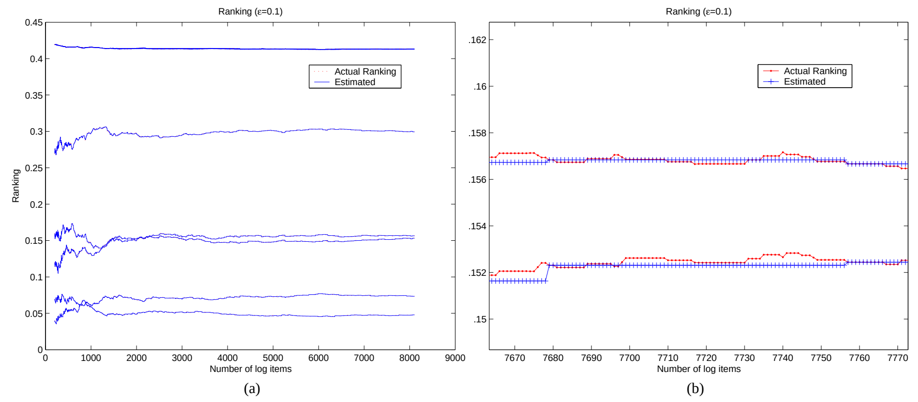

Figure 4.3. Rankings of "top" 10 users. (b) enlarges part of (a) for clarity.

data accumulates and allows for more and more perturbation while maintaining the given precision. Figure 4.2(b) enlarges one area of (a) to show the details. This area lies between data item 5965 to 6078. The horizontal line segments of red dots represent the intervals where the perturbation is within tolerable range and no update is applied. This particular line in Figure 4.2(b) demonstrates around 113 updates for which the NewHITS needs not to be invoked, i.e., a saving of 113 rounds of HITS computation. Similar savings can also be observed in other areas of Figure 4.2(a).

Figure 4.3 shows the rankings of "top" 10 users in the data set [4]. Both the actual

[4]For privacy reasons the names of the users are withdrawn.

ranking of each user (obtained by running HITS at each update) and the approximation produced by OnlineHITS are plotted. Note that in Figure 4.3(a) the rankings of the top 5 users are so close that their results appear in the figure almost as a single curve (the curve on the top). Preliminary investigation uncovered that they are all involved in a large number of error messages (one of them is the sender and the rest recipients) and, as the HITS algorithm discovered, they share similar roles in terms of their email communication pattern in the data set. Our algorithm discovers this structure as well. The estimated rankings are so close to the actual ones that it is difficult to distinguish them in Figure 4.3(a). Figure 4.3(b) enlarges part of (a) for clarity. It shows that the estimated rankings closely track the actual ones even when no recomputation is performed.

Our test demonstrates the substantial advantage of OnlineHITS when applied to an actual data corpus. We believe it is applicable to other dynamic environments as well. In particular, for systems that do not have a clearly marked leisure period (e.g. a system serving users from all time zones around the world), simply "running HITS at night" will not work. Our algorithm can provide an accurate estimate on the perturbation a update can cause and offers precise ranking in real-time.

4.7 Privacy Preserving Online HITS

The algorithm described in previous sections addresses the dynamic and real-time response issues of using access patterns in link analysis. However, in many situations, a naive implementation of the algorithm has severe privacy implications. In most applications, the weight on each edge of the graph represents the "rating" or "preference" of a user to the documents or other user and is gathered via client side logging. Such information is quite personal and exposing it would jeopardize the privacy of users thus hindering the acceptance of our system. If implemented directly, the online HITS algorithm would require the server running the algorithm be able to see all the data and involve substantial amount of network communication. In most situations trusting the server or network is not acceptable.

This problem can be solved with our P4P framework. In the following we will only consider the computation of document ranking, x. Expertise ranking is done in a similar fashion.

4.7.1 A Run of HITS

The results of the tth iteration of HITS, x^t and y^t which are aggregate data, are made public. User i is responsible for his own rating of the documents (obtained via analyzing his document access pattern), namely the ith row of matrix A. Let $A_i^T = [a_{i1}, a_{i2}, \ldots, a_{im}]$ be that row. For the step $x^{t+1} = A^T y^t$, all that is involved from i is his own expertise ranking, y_i^t, and A_i^T. User i computes $y_i^t A_i^T$ and inputs the vector to the P4P computation which computes and broadcasts the aggregate. This is x^{t+1}.

To compute y^{t+1} (which is Ax^{t+1}), user i computes $A_i^T x^{t+1}$ which is y_i^{t+1}, the ith element of vector y at iteration $t+1$, and publish it. If every user does this, the vector y^{t+1} can be obtained.

The iteration can stop when enough precision is achieved.

4.7.2 Perturbation Checking

The scheme described in Section 4.7.1 shows a run of HITS, not Online HITS. To fit online HITS into such a scheme, we need to find a way to compute the perturbation, $\|E_S(t)\|_F$, with encrypted data or allow each user to compute with local data.

Recall that Equations 4.7, 4.8, 4.9 and 4.10 give us a way to update the upper bound of $\|E_S(t)\|_F$. The terms that need to be computed for each update are $\|A^T\Delta(t)\|_F$, $\|E(t-1)^T\Delta(t)\|_F$ and $\|\Delta(t)^T\Delta(t)\|_F$. Since for user i, $\Delta(t)$ has non-zero elements only in its ith row (and these numbers are obtained locally via his document access pattern analysis), $A^T\Delta(t)$ only involves the ith row of A, which the user maintains. Similarly, $E(t-1)^T\Delta(t)$ only involves the ith row of $E(t-1)$. In short, each user's update only involves his local data and it is straightforward to perform perturbation checking without disclosing private data: $\|E_S(t)\|_F$, $\|A^T E(t)\|_F$ and $\|E(t)^T E(t)\|_F$ are made public and each user will update them using Equations 4.8, 4.9 and 4.10 with their local updates. When it is determined that it is time to update A, each user will update his own row and reset the perturbation records. All of them then collaboratively run the HITS algorithm as described in Section 4.7.1.

Note that we have actually killed two birds with one stone if we perform perturbation checking this way. Not only could we preserve user's privacy, we also distributed the computation among all users and parallelized the process.

4.8 Related Work

In [110], a set of heuristic graph algorithms are used to uncover shared-interest relationships among people, and to discover the individuals with particular interests and expertise, based on the logs of email communication between these individuals. The limitation with this approach is that experts are assumed to be communicating with fellow experts, which is not necessarily true in the real-world where experts may not be acquainted with one another, or may be rivals. Our approach does not assume any particular communication patterns between experts, and instead locate the experts based on their activities, e.g. if an expert accesses this set of authoritative documents, another person who accesses the same set is likely to be an expert as well.

The Referral Web [73, 74] is an interactive system for restructuring, visualizing and searching social networks on the World Wide Web. It constructs a graph of all users based on their email communication logs, which it uses to send a chain of referral requests until these requests reach an expert user. Like our Online HITS algorithm, Referral Web constructs the social network incrementally as it indexes the documents created and received by users. In contrast to our approach, however, the Referral Web raises possible privacy concerns because the chain of referrals inevitably reveal who someone down the chain knows to the user who initiates the search, unless individuals down the chain chooses not to forward the referral, in which case it becomes harder for the query to succeed.

Pirolli et al. [103] use a link-based approach like HITS to categorize webpages. It is similar to our weight-based algorithm in that users' access paths and metadata about webpages are used to construct the appropriate matrices. It differs significantly from ours in that while we use successive iterations to converge on our results, Pirolli et al. construct an activation network based on the strength of association between webpages and use the spread of activation in this network, starting from identified source webpages, to identify the webpages that exceed a threshold quantity of flow.

Carriere and Kazman's WebQuery system [24] rank webpages by considering the number of neighbors in the hyperlink structure that each webpage has. WebQuery performs link-based query post-processing to improve the quality of the results that it returns. In contrast, our incremental approach assumes that the hyperlink structure is highly dynamic, and postpones processing until the latest user-document accesses accumulate significant perturbation.

4.9 Conclusion

We extended the HITS hyperlink analysis algorithm to make it applicable to analyzing dynamic weighted graphs. Our generalizations are in two directions. First, we replaced the 0-1 valued connectivity property with a non-negative valued weight function to allow for encoding of richer information. We proved the convergence HITS on such weighted graphes. Second, we created an online version of the HITS algorithm that can approximate the results efficiently in face of frequent updates by estimating the upper bound of perturbation and postponing applying the updates whenever possible. Both theoretical analysis and empirical experiments show that our generalized online algorithm is more efficient than the original HITS under the context of dynamic data.

Finally we developed a secure and distributed implementation of our online algorithm that solves the potential privacy issues that may occur when deploying large-scale access pattern-based document and authority ranking systems. Our scheme makes use of cryptographic techniques such as threshold decryption and homomorphic public-key encryption and distributes computation among users. Only aggregate or encrypted data are exposed. The scheme is also robust against a number of dishonest users up to a certain threshold.

Our extensions to Kleinberg's original HITS algorithm result in a generalized algorithm, Secure OnlineHITS, that is practical for link analysis in scenarios such as collaborative work and online communities, in which there is no explicit link structure among nodes, and that users' access patterns of documents are highly dynamic, complex and should remain private.

Chapter 5

Association Rule Mining

The association rule mining [2] is still one of most popular pattern-discovery methods in the field of knowledge discovery. Briefly, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items. The meaning of such rules is as follows: Given a database D of records, $X \Rightarrow Y$ means that whenever a record R contains X then R also contains Y with certain confidence. The rule confidence is defined as the percentage of records containing both X and Y with regard to the overall number of records containing X. The fraction of records R supporting an item X with respect to database D is called the support of X.

5.1 Distributed Association Mining

Let n be the total number of users and m be the total number of items. User i maintains a private data set D_i, $i = 1, 2, \ldots, n$ (which may contains e.g. her purchase records). The data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ forms a database, which is actually the concatenation of D_1, D_2, $\cdots$ and D_n. The database can be represented by a $n \times m$ boolean matrix x where $x_{ij} = 1$ if record i contains item j and 0 otherwise. We consider horizontally partitioned database [134] where each user's data set contains the same attributes. Without causing confusion, we also use D_i to denote the rows of the matrix x maintained by user i.

The goal is to conduct association rule mining on $[D_1 \cup D_2 \cup \cdots \cup D_n]$ and to find the association rules with support and confidence being greater than the given thresholds. We say an association rule (e.g., $X \Rightarrow Y$) has confidence $c\%$ in the data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ $c\%$ of the records which contain X also contain Y (namely, $c\%$ $= P(Y \mid X)$). We say that the association rule has support $s\%$ in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if $s\%$ of the records in $[D_1 \cup D_2 \cdots \cup D_n]$ contain both X and Y (namely, $s\% = P(X \cap Y)$). Consequently, in order to learn association rules, one must compute the candidate itemsets, and then prune those that do not meet the preset confidence and support thresholds. In order to compute confidence and support of a given candidate itemset, we must compute, for a given itemset C, the frequency of attributes (items) belonging to C in the entire database (i.e., we must count how many attributes in C are present in all records of the database, and divide the final count by the size of the database which is m.) Note that association rule mining works on binary data, representing presence or absence of items in transactions. However, the proposed approach is not limited to the assumption about the binary character of the data in the content of association rule mining since non-binary data can be transformed to binary data via discreterization.

5.2 Association Rule Mining Procedure

The following is a fast algorithm for mining association rules on $[D_1 \cup D_2 \cdots \cup D_n]$, introduced in [3]:

```
1. $L_1$ = large 1-itemsets
2. **for** $(k = 2; L_{k-1} \neq \emptyset; k++)$ **do begin**
3.    $C_k$ = **apriori-gen**$(L_{k-1})$
4.      **for** all candidates $c \in C_k$ **do begin**
5.        **Compute *c.count***
6.      **end**
7.    $L_k = \{c \in C_k | c.count \geq \textit{min-sup}\}$
```

```
8. end
9. Return L = $\cup_k L_k$
```

At line 5 in the above procedure, *c.count* divided by the total number of records is the support of a given item set. We will show how to compute it in Section 5.3.

The procedure **apriori-gen** is described in the following (please also see [2, 3] for details).

```
apriori-gen($L_{k-1}$: large $(k-1)$-itemsets)

    insert into $C_k$
    select $p.item_1$, $p.item_2$, ..., $p.item_{k-1}$,$q.item_{k-1}$
    from $L_{k-1}$ $p$, $L_{k-1}$ $q$
    where $p.item_1 = q.item_1$, ..., $p.item_{k-2} = q.item_{k-2}$
            $p.item_{k-1} < q.item_{k-1}$;
```

Next, in the *prune* step, we delete all itemsets $c \in C_k$ such that some $(k-1)$-subset of c is not in L_{k-1}:

```
∀ itemsets $c \in C_k$ do
    ∀ $(k-1)$-subsets $s$ of $c$ do
        if($s \notin L_{k-1}$) then
            delete $c$ from $C_k$;
```

5.3 Computing *c.count*

In the procedure of association rule mining for horizontal partition, each party computes the partial *c.count* based on their own data. Without loss of generality, let's assume that user i maintains $c.count_i$, $i = 1, \dots, n$. The goal is to compute the $c.count$ which equals to $\sum_{i=1}^{n} c.count_i$ without disclosing $c.count_i$ to party j where $i \neq j$. In the next section, we will provide a privacy-preserving protocol to compute this summation under P4P framework.

5.4 Association Rule Mining in P4P

Algorithm **apriori-gen** is to generate a superset of possible candidate itemsets and then prune it to get C_k. We use the P4P vector aggregation and verification protocols to compute $c.count$ for all $c \in C_k$ as one P4P aggregation step.

Recall that D_i is the data set of party i. In the case of horizontal partition, D_i is essentially a number of rows of items. At step k of the above algorithm, let $m_k = |C_k|$ and $C_k = \{c_1, \dots, c_{m_k}\}$. The protocol is as follows:

1. User i constructs an m_k-dimensional vector $d_i^{(k)} \in \mathbb{Z}_\phi^{m_k}$ such that

$$d_i^{(k)}[j] = \sum_{r \in D_i} (\prod_{l \in c_j} x_{rl})$$

2. User i then inputs $d_i^{(k)}$ into the P4P protocol. She also provides a zero-knowledge proof that her data is valid using the technique that will be elaborated later. If the user fails the proof, the server and the privacy peers exclude her data from subsequent computation.

3. Let $s^{(k)} = \sum d_i^{(k)}$ be the output of the P4P aggregation. The desired counts are simply $c_j.count = s^{(k)}[j], j = 1, \dots, m_k$.

At the first step, for all the rows in her data set, for each itemset $c_j \in C_k$, the user computes the product of all the attributes in c_j. She then add these products across all the rows in her block. This is the j-th elements in d_i. Clearly, this is the number of rows in her data set containing the itemset c_j. By aggregating these vectors across all users, P4P outputs the total number of rows containing all itemsets $c_j \in C_k$ in the entire database $[D_1 \cup D_2 \cup \cdots \cup D_n]$.

At step 2, the user is required to prove, in zero-knowledge, that her data is valid. There are two ways to accomplish this that can be used in different situations. When k is small, there hasn't much pruning yet so the candidate set C_k tends to be quite large. For typical data such as web pages or vendor inventory, m_k can be in the range of billions or millions. In this case the L2-norm verification protocol introduced in chapter 2 should be used with a bound

$$L = \alpha\sqrt{m_k}|D_i|$$

where $0 < \alpha \leq 1$.

At the later steps of the algorithm when k is large, C_k can become small since it has gone through many iterations of pruning. In this case one can apply the ZK boundedness proof directly on each of d_i's elements to verify that all of them are bounded by $|D_i|$. Please see chapter 2 and [15] for details.

5.5 Discussion

5.5.1 Privacy Analysis

The privacy of our solution is straightforward to analysis. As is proven in chapter 2, the P4P aggregation and validation do not leak any more information about individual user data other than what can be inferred from the final results. And the results are just the supports of the itemsets that we are computing. These have been treated as public data by many private association rule mining schemes (e.g. [122, 135]).

5.5.2 Near Optimal Efficiency

In a realistic association rule mining application, the scale of the problem is typically quite large. For example, WalMart sells over 100,000 items. And the web has billions of web pages. When mining association rules using the a priori algorithm over such datasets, C_k may contain millions of itemsets. For any solutions to be practical, they must be efficient enough to handle dataset of such scale.

The P4P solution is *near optimal* in that its cost is comparable to that of a distributed, non-private implementation of the a priori algorithm. For each iteration, computing *c.count* for all $c \in C_k$ is simply one single P4P vector aggregation. And since the P4P's main computation (which has complexity $O(m_k|D_i|)$ for party i) is done over small field (32- or 64-bit) and has the same cost as regular, non-private arithmetic, compared to a server-based, non-private implementation of the algorithm, the computation overhead for each user in P4P is only doubled while that for the server remains the same. Verification is done in large field but the number of such operations is only $O(\log m_k)$. This is one of the important features provided by the P4P framework and it guarantees that the privacy mechanism causes little overhead for the server or the privacy peers, which are the bottlenecks of the system. Using the benchmark obtained by the P4P implementation in [23], verification of a million element vector takes only a few seconds of user and server time. The same task takes hours using other solutions. In a sense P4P provides privacy to association rule mining applications almost for free.

5.5.3 Dealing with Malicious Users

A lot of private data mining schemes such as [132] cannot deal with active cheating from users (in cryptographic terms, their schemes are only secure passive adversary). However, in any realistic applications, users often pose active threats. A user may submit invalid data, or refuse to participate in the computation at all. A competitor of the data mining service provide can easily disguise as a user and disrupt the computation.

In our scheme we use P4P's efficient built-in zero-knowledge user data validation protocol to verify that a user cannot exert too much influence on the computation. The bound used in the protocol should be at most $\sqrt{m_k}|D_i|$ (corresponding to the case of setting $\alpha = 1$), but usually substantially smaller, determined by the application based on the estimate on the distribution of the raw data. The check is not tight in that a cheating user can still falsify her data without being detected. However, the effectiveness of her cheating is limited (by the bound) and assuming a reasonably large fraction of users are honest (otherwise there is no point running the data mining algorithm anyway), cheating users cannot cause too much change to the result.

In addition, our solution features a simple cheating user handling mechanism. When a user is detected cheating by P4P's validation protocol, we simply disqualify her from the computation for future steps. This is equivalent to setting all her entries to 0s for the rest of the computation. The results from previous steps may not be very accurate but generally there is no need to redo them. This is because the validation ensures that if a user tries to manipulate the support of an itemset by increasing her data in the corresponding entry by a large amount, she will be detected. So the only undetectable cheating is to increase the value by a small amount or decrease it. If the user only holds a few rows of the database, which typically should be the case for typical user-based mining, neither should have big influence on the computation since we are aggregating across a large number of users and most of them are honest. If, on the other hand, she holds a large trunk of the database, the server and the privacy peers can simply exchange the shares of her data and reconstruct her original vector $d_i^{(1)}$. They then correct the results by subtracting all cheating users data from the aggregation. The correction is done in public, in regular-sized field and should be very efficient.

Chapter 6

An IND-CCA2 Multicast Cryptosystem

The P4P framework and many network applications are based on a group communications model where one party sends messages to a large number of authorized recipients and/or receives messages from multiple senders. In this chapter and next, we present some scalable and secure communication primitives that are suitable for P4P's communication patterns. This chapter introduces a general framework for constructing efficient multicast cryptosystems with provable security. We show that a line of previous work on multicast encryption are all special cases of this general approach. We provide new methods for building such cryptosystems with various levels of security (e.g., IND-CPA, IND-CCA2). The results enable the construction of a whole class of new multicast schemes with guaranteed security using a broader range of common primitives such as OAEP. Moreover, we show that multicast cryptosystems with high level of security (e.g. IND-CCA2) can be based upon public key cryptosystems with weaker (e.g. CPA) security as long as the decryption can be securely and efficiently "shared". The constructions feature truly constant-size decryption keys whereas the lengths of both the encryption key and ciphertext are independent of group size.

6.1 Secure Multicast

Multicast offers an efficient way to deliver the same message to a group of receivers and has become the basis of many applications. The Internet today supports a basic form of multicast service. On the Internet, a multicast group is identified by a Class D IP address and any receivers can join or leave a multicast group by sending IGMP (Internet Group Management Protocol) [51] messages to their local router. Any sender can send message to a multicast group by addressing the message to the group address.

The current IP Multicast service does not provide mechanisms to restrict message delivery to a specified set of receivers therefore other means have to be used to secure the communication. A multicast encryption system provides *confidentiality* for multicast data – ensuring that any parties other than the intended recipients should not be able to access the message. To this end, most of the existing work use one of two approaches. The first is represented by the work in network research that is concerned with *multicast security*. In this approach symmetric key encryption is used and the data is encrypted with a traffic encryption key (TEK) that is known only to the multicast group members. The difficulty here is key management: The TEK may have to be changed when members join or leave the group. This is known as re-keying. Early schemes (e.g., Group Key Management Protocol (GKMP) [66]) let the group controller or the sender share a pairwise key with each group member and distribute keys to them on a one-to-one basis. For obvious reasons this cannot scale to large groups.

Some work has been done to improve the scalability of such schemes. Among the efficient solutions, the Logical Key Hierarchy (LKH) (or Key Graph) was independently discovered in [125] and [128] and has been an inspiration for many subsequent works [19, 26, 129, 81, 111, 131]. In these schemes, individual and auxiliary keys are organized into a hierarchy and each group member is assigned to a leaf and holds all the keys from its leaf to the root. The root key is shared by all group members and used as the TEK. New TEK is distributed by encrypting it with keys that deleted members do not have. So far $O(\log n)$ seems to be the best storage (for both center and members) and communication complexity the LKH-based schemes achieved, where n is the size of the multicast group.

The problem with this approach is that revoking a single user involves changing the keys for all others and the receivers must be stateful and always online in order to receive the latest TEK.

The second approach uses asymmetric key cryptosystem and allows the receivers to be stateless. This includes the work in cryptography such as *traitor tracing*, a concept introduced by Chor, Fiat and Naor [27], and *broadcast encryption*, initiated by Fiat and Naor [52]. Both are based on encryption schemes where a ciphertext can be decrypted by multiple parties with different keys. The scheme in [52] requires $O(t \log t \log n)$ keys per user and the transmission of $O(t^2 \log^2 t \log n)$ messages where t is the number of revoked users. Subsequent work proposed a number of other schemes including [13, 5, 85, 56, 93], and [65, 91] which achieved $O(t)$ message complexity and $O(\log^{1+\epsilon} n)$ keys per user. Boneh and Franklin's scheme proposed in [13] is based on Reed-Solomon codes and the representation problem for discrete logs. They also presented a modification, using techniques by Cramer and Shoup [35], that was provably secure against adaptive chosen ciphertext attack.

Recently Boneh et al. presented a broadcast encryption scheme based on bilinear map with constant-size ciphertexts and private keys (and $O(n)$-size public key) [14]. However, in this system, the decryption requires the public key and the knowledge of the set of legitimate recipients. Therefore the "effective" decryption key and/or ciphertext in a real application actually become linear in the total number of receivers.

There is a line of work in the second approach that we classify as Asymmetric Threshold Decryption-based (ATD-based) multicast encryption. This includes [5, 93, 121, 44, 76], although none of them explicitly formalized their schemes this way. In these schemes a private key is shared using a $(t + 1, n + t)$-threshold scheme and the shares are distributed *asymmetrically.* Namely the center is given t shares and each user is given 1 share. The center broadcasts a ciphertext together with t partial decryptions. Any member with a valid share of the private key can produce another decryption share and recover the message.

With such schemes, user only needs to store a key of constant length. And both the message complexity and sender storage are $O(t)$, independent of the group size.

6.1.1 Our Results

We focus on the ATD-based multicast encryption cryptosystems and introduce a general framework for constructing such systems with guaranteed security. As we will show later, all existing ones are special cases of our constructions. In particular, they are all based on specific ElGamal encryption that relies on specific assumptions (e.g. DDH). The results we obtained in this chapter, on the other hand, are more general. The main contributions are: (1) We show that *any* threshold encryption scheme can be used to construct a multicast cryptosystem that retains the same level of security (e.g. IND-CPA, IND-CCA2) as the underlying threshold encryption. (2) We obtain new results that improve over existing ATD-based schemes in both security and efficiency. Specifically, the resulting scheme from our construction can be made CCA-secure even if the underlying threshold scheme is not. (3) Furthermore, we show that an IND-CCA2 secure multicast scheme can be constructed from a public key cryptosystem that does *not* have a secure threshold implementation (such as OAEP) or has only weaker security (e.g. only IND-CPA), provided the decryption can be securely and efficiently shared (to be elaborated in Sect. 6.4.4). All of our security proofs are in the same (standard or random oracle) model as the underlying threshold scheme or public key cryptosystem.

These general security results can be used to analyze existing systems in a more unified framework and provide guidelines for constructing future schemes with guaranteed security. This frees the system designer from the burden of security consideration and allows them to focus on other aspects of their schemes.

6.2 Preliminaries

We consider the scenario where a single party, called *the center*, sends messages, over insecure channels, to a group U of n parties who are denoted *members* of the group. In such a setting, the center often has a special role. Since it is often distributing information of its own choice, it is assumed to have control over the group membership, i.e., the center is allowed to make decisions about who can join the group and whose membership should be revoked. This is in line with almost all multicast schemes such as [84, 128, 117, 126, 121, 44, 76].

We assume a computationally bounded adversary who is allowed to attack the system from both outside and *inside* the group. The insider's attack is modelled by allowing the adversary to corrupt and gain total control of up to t group members where t is a predefined threshold. We only consider non-adaptive adversary who chooses what members to corrupt before the key generation.

The multicast communication we are considering in this chapter is assumed to be "closed", i.e., we only provide *the center* with the ability to encrypt messages (and of course only the intended recipients can decrypt them). This is different from the public key systems such as [93, 121, 44, 76] where the information to encrypt a message is public. The openness is unnecessary for some applications and unacceptable for some others (e.g. military communication). By "closing" the communication, we can provide more flexible constructions that can make use of a broader range of primitives. The price for this flexibility is the loss of the public key feature, which should not be a problem for many applications. However we observe that in many instantiations of our constructions, it is easy to "publicize" the encryption key, without affecting the security of the scheme, as demonstrated by works such as [121, 44, 76]. This effectively turns the scheme into a public key system and all the openness features are reinstalled.

6.3 Multicast Cryptosystem

Definition 1. *An n-way multicast encryption scheme $\mathcal{ME} = (\mathsf{KeyGen}, \mathsf{Reg}, \mathsf{E}, \mathsf{D})$ consists of the following set of algorithms:*

1. *Key Generation* KeyGen*: a probabilistic polynomial-time (in k) algorithm which takes*

as inputs a security parameter 1^k, a threshold t, the number of (initial) group members n, and generates global information I, the encryption key Σ and the master secret key Γ.

2. *Registration algorithm* Reg*: a probabilistic algorithm to compute the secret initialization data for a new user subscribing to the system.* Reg *receives as input the master key Γ and a new index i associated with the user; it returns the user's secret key Γ_i.*

3. *Encryption* E*: a probabilistic polynomial-time algorithm that, on inputs Σ, the encryption key, and a string $m \in \{0,1\}^k$, and a set R of revoked users (with $|R| \leq t$) and their keys, produces as output $\psi \in \{0,1\}^*$ called the ciphertext* [1].

4. *Decryption* D*: a deterministic polynomial-time algorithm such that $\forall\, m \in \{0,1\}^k$, $\forall\, i \in U \setminus R$, $\mathsf{D}(\Gamma_i, \mathsf{E}(\Sigma, \{(j, \Gamma_j) | j \in R\}, m)) = m$. On all other inputs it outputs a special symbol $\perp$.*

KeyGen and Reg should be run by the center and the two can also be executed together with an initial set of n members as input. Admitting new members is relatively trivial, at least for all the construction we will be presenting, so in the following we simply omit Reg and use $(I, \Sigma, \Gamma, \vec{\Gamma}) \leftarrow \mathsf{KeyGen}(1^k, t, n)$ to denote this process, where $\vec{\Gamma} = (\Gamma_1, \ldots, \Gamma_n)$ is a vector of secret keys for the n members.

6.3.1 Notion of Security

The communication paradigm we are considering shares similarities with both symmetric key and public key cryptosystems. On one hand the communication is "closed" in that we only allow the center to send messages to the group. On the other hand the keys are "asymmetric" since now there are multiple recipients and our definition includes member revocation which means the encryption key and the decryption keys must be different.

Dodis and Fazio [44] first precisely formalized the notion of adaptive security for public key multicast encryption schemes, which allow anyone having access to the public key to

[1] *Note that member revocation is implicitly embedded in the encryption algorithm.*

send messages to the group, at both CPA and CCA2 levels. Since our setting is different from the "public key" paradigm, we adopt a slightly modified definition. The major difference is that, we do not *explicitly* allow the adversary to see the sender's keys since ours is not a public key cryptosystem. Instead the adversary can obtain encryptions of arbitrary messages by querying an *encryption oracle* who also encrypts the target message later. This is similar to the security definition based on indistinguishability for symmetric key cryptosystems. The ability to handle member revocation is modelled by allowing the adversary to corrupt members and obtain their secret keys. This formalization is general and captures the security notions of many multicast schemes such as those LKH schemes [125, 128] which are based on symmetric key cryptography. However we note that in all the construction we introduce later, the secret keys of the revoked members constitute the actual *encryption key*. In essence in our constructions the exposure of encryption key can be modelled as corrupting members. This effectively turns our scheme into a "public key" paradigm from the adversary's point of view and the security definitions from [44] are appropriate.

Formal Model

Given a multicast encryption scheme $\mathcal{ME} = (\mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$, a polynomial time adversary $\mathcal{A}$'s attack is modelled by the following game:

Game ME:

M1 The adversary $\mathcal{A}$ chooses to corrupt a fixed set R of t members.

M2 $(I, \Sigma, \Gamma, \vec{\Gamma}) \leftarrow \mathsf{KeyGen}(1^k, t, n)$ is run and $\mathcal{A}$ is given the public information I and the secret keys of corrupted members. User i receives Γ_i. The center is given R and their keys.

M3 The adversary interacts with the center, who acts as the encryption oracle, in an arbitrary fashion. On any query m from $\mathcal{A}$, the center returns its encryption.

M4 $\mathcal{A}$ chooses two plaintexts m_0 and m_1 of the same length and gives them to the center who chooses $b \in \{0,1\}$ at random, and gives the "target" ciphertext $\psi' = \mathsf{E}(\Sigma, \{(j, \Gamma_j) | j \in R\}, m_b)$ to $\mathcal{A}$.

M5 $\mathcal{A}$ continues to interact with the center.

M6 At the end of the game, $\mathcal{A}$ outputs $b' \in \{0,1\}$.

The advantage of $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{CPA}_{\mathcal{ME},\mathcal{A}}(k) = |Pr(b' = b) - 1/2|$$

In addition, in the case of a adaptive chosen ciphertext attack (CCA2) [2], in both stages M3 and M5, $\mathcal{A}$ is also allowed to interact in an arbitrary manner with the group members who act as the decryption oracles. On a query ψ from $\mathcal{A}$, member i returns $\mathsf{D}(\Gamma_i, \psi)$. The only restriction on the interaction is that the target ciphertext ψ' cannot be one of the queries made to any of the decryption oracles. As before, $\mathcal{A}$'s advantage in the CCA2 case is defined as

$$\mathsf{Adv}^{CCA2}_{\mathcal{ME},\mathcal{A}}(k) = |Pr(b' = b) - 1/2|$$

Definition 2 (t-Resilient Multicast Encryption Scheme). *Let $\mu \in \{CPA, CCA2\}$. A multicast encryption scheme $\mathcal{ME}$ is t-resilient against a μ-type attack if the advantage, $\mathsf{Adv}^{\mu}_{\mathcal{ME},\mathcal{A}}(k)$, of any probabilistic polynomial time adversary $\mathcal{A}$ is a negligible function of k.*

6.4 ATD-Based Multicast Encryption

In this section we define two constructions and show that a line of previous work on multicast or broadcast encryption can actually be characterized as special cases of these constructions.

[2]We do not explicitly consider non-adaptive chosen ciphertext attack (CCA1). It should be easy to see that all the discussions and proofs still hold in the case of CCA1, by simply restricting the adversary from interacting with the decryption oracles after the target ciphertext is generated in both Game ME and TD (Sect. 6.4.1).

6.4.1 Threshold Decryption Scheme

A $(t+1,n)$-threshold cryptosystem $\mathcal{TD} = (\mathsf{KeyGen_{TD}}, \mathsf{D_{TD}}, \mathsf{V_{TD}}, \eta, \mathsf{E_{TD}})$ consists of the following algorithms:

- Key generation algorithm $(PK, VK, \vec{SK}) \leftarrow \mathsf{KeyGen_T}(1^k, t, n)$: a probabilistic algorithm that, given a security parameter 1^k, a threshold t, and the number of players n, generates a public key, PK, a verification key VK, and n private keys $\vec{SK} = (SK_1, \ldots, SK_n)$. PK and VK are made public while SK_i is known only to player i, $i = 1, 2, \ldots, n$.

- Share computation $\mathsf{D_{TD}}$: a probabilistic algorithm that, given a private key SK_i and the ciphertext c, $\mathsf{D_{TD}}$ computes $\rho = \mathsf{D_{TD}}(SK_i, c)$, called a *decryption share*.

- Share verification V: a deterministic algorithm that takes as input the public verification key VK, the ciphertext c, and a share ρ, and outputs $\mathsf{V}(VK, c, \rho) \in \{0, 1\}$.

- Share combination algorithm η: given the verification key VK, the ciphertext c, and a set Λ of $t+1$ shares, η either outputs the corresponding result $r = \eta(VK, c, \Lambda)$ or a special symbol $\perp$ that is different from all possible correct results.

- Encryption algorithm $\mathsf{E_{TD}}$: the "opposite" of $\mathsf{D_{TD}}$. This function is carried out in the normal manner by a single party and should follow the same definition as the encryption algorithm in a standard public key cryptosystem.

The operation of a threshold decryption scheme can be modelled as follows. There is a trusted dealer (e.g. the center) and a set of n decryption servers indexed $1, \ldots, n$. In an initialization phase, the dealer runs the key generation algorithm and creates PK, VK and $\vec{SK}$. SK_i is given to server i. To decrypt a ciphertext ψ, a client gives ψ to the servers, requesting a decryption share from each of them. It can verify the validity of the shares using the given verification key. Once the client collects valid shares from at least $t+1$ servers, she can apply η to obtain the decryption.

Threshold cryptosystems are part of a general approach known as threshold cryptography, introduced by Boyd [16], Desmedt [41], and Desmedt and Frankel [42]. There are schemes based on both Diffie-Hellman problem [42] and RSA [38]. All these schemes can be shown to be secure against chosen plaintext attack, but they are not known to withstand chosen ciphertext attack. After Cramer and Shoup discovered the first truly practical public key cryptosystem that is provably secure against chosen ciphertext attack *without* using random oracles [35], several of its threshold implementations have been proposed and proved CCA2 secure (also without using the random oracle model) [20, 1, 70]. Shoup and Gennaro presented a more efficient threshold scheme in [116] that is proven CCA2 secure in the random oracle model.

We adopt Shoup and Gennaro's definition of security for threshold decryption schemes from [116], which is a natural extension of security for a public key cryptosystem, and define the security of a $(t+1, n)$-threshold decryption scheme $\mathcal{TD} = (\mathsf{KeyGen}_{\mathsf{TD}}, \mathsf{D}_{\mathsf{TD}}, \mathsf{V}_{\mathsf{TD}}, \eta, \mathsf{E}_{\mathsf{TD}})$ with respect to the following game:

Game TD:

TD1 The adversary $\mathcal{A}$ chooses to corrupt a fixed set of t servers.

TD2 The key generation algorithm is run. The public key, verification key and the private keys of the corrupted servers are given to $\mathcal{A}$. Other private keys are given to the uncorrupted servers.

TD3 $\mathcal{A}$ chooses two plaintexts m_0 and m_1 of the same length and gives them to an "encryption oracle" that chooses $b \in \{0, 1\}$ at random, and gives the "target" ciphertext $\psi' = \mathsf{E}_{\mathsf{TD}}(PK, m_b)$ to $\mathcal{A}$.

TD4 At the end of the game, the adversary outputs $b' \in \{0, 1\}$.

This game defines the attack scenario for CPA security. The adversary's advantage is defined to be the absolute difference between 1/2 and the probability that $b' = b$:

$$\mathsf{Adv}^{CPA}_{\mathcal{TD},\mathcal{A}}(k) = |Pr(b' = b) - 1/2|$$

For CCA2 attacks, $\mathcal{A}$ is allowed to interact with uncorrupted decryption servers, who act as the decryption oracles, in an arbitrary fashion, feeding them ciphertexts $\psi \neq \psi'$, and obtaining decryption shares. The calls to the decryption oracles can happen at any point during the execution of the game, both before and after stage TD3, and be arbitrarily interleaved with other oracle calls. $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}^{CCA2}_{\mathcal{TD},\mathcal{A}}(k) = |Pr(b' = b) - 1/2|$$

Definition 3 (t-Resilient Threshold Decryption Scheme). *Let $\mu \in \{CPA, CCA2\}$. A threshold decryption scheme $\mathcal{TD}$ is t-resilient against μ-type attacks if the advantage, $\mathsf{Adv}^{\mu}_{\mathcal{TD},\mathcal{A}}(k)$, of any probabilistic polynomial time adversary $\mathcal{A}$ is a negligible function of k.*

6.4.2 Basic Construction

Construction 1 (ME1). *Given a threshold decryption $\mathcal{TD} = (\mathsf{KeyGen_{TD}}, \mathsf{D_{TD}}, \mathsf{V_{TD}}, \eta, \mathsf{E_{TD}})$, a security parameter 1^k, a threshold t and the number of (initial) members n, a multicast encryption scheme $\mathcal{ME}^{\mathcal{TD}}_{C1} = (\mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ can be constructed as follows:*

1. *Key Generation KeyGen: Run $(PK, VK, \vec{SK}) \leftarrow \mathsf{KeyGen_{TD}}(1^k, t, n+t)$. Set $I = (PK, VK)$ and the encryption key $\Sigma = \{(j, SK_j) : j = n+1, \ldots, n+t\}$. Σ is given to the center. Member i receives secret key $\Gamma_i = (i, SK_i)$. The master secret key is $\vec{\Gamma} = (\Gamma_1, \ldots, \Gamma_{n+t})$.*

2. *Encryption E: Given a set R of revoked members, and their secret keys, with $|R| \leq t$, a message m, the encryption proceeds as follows. Let $T = \{n+1, \ldots, n+t\}$. The encryptor randomly selects a subset of T with $t - |R|$ elements, denoted T', and computes the ciphertext $\psi = (c, \{(j, c_j) : j \in T' \cup R\})$ where $c = \mathsf{E_{TD}}(PK, m)$ and $c_j = \mathsf{D_{TD}}(SK_j, c)$.*

3. *Decryption D: Given a secret key Γ_i and a ciphertext ψ, the ciphertext is first parsed into $\psi = (c, \Lambda')$ where $\Lambda' = \{(j, c_j) : j \in T' \cup R\}$ with $c_j = \mathsf{D_{TD}}(SK_j, c)$. For all*

$j \in T' \cup R$, the decryption first test $v_j = \mathsf{V}_{\mathsf{TD}}(VK, c, c_j)$. If any $v_j = 0$, D returns $\perp$. Otherwise it returns

$$m = \eta(VK, c, \Lambda' \cup \{(i, \mathsf{D}_{\mathsf{TD}}(SK_i, c))\}) \tag{6.1}$$

With this construction, the multicast ciphertext essentially consists of the ciphertext of the underlying threshold scheme, together with t partial decryptions produced using the keys of revoked members. To decrypt, a legitimate member combines the partial decryptions embedded in the ciphertext with another one computed using her own share of the private key. As we will show, this construction preserves the security of the underlying threshold scheme.

Theorem 6 (Security Inheritance). *Let $\mu \in \{CPA, CCA2\}$. Given a threshold decryption scheme $\mathcal{TD} = (\mathsf{KeyGen}_{\mathsf{TD}}, \mathsf{D}_{\mathsf{TD}}, \mathsf{V}_{\mathsf{TD}}, \eta, \mathsf{E}_{\mathsf{TD}})$ that is t-resilient against μ-type attacks, the multicast encryption scheme $\mathcal{ME}_{C1}^{\mathcal{TD}}$ constructed using Construction 1 with threshold t and (initial) group size n is t-resilient against μ-type attacks.*

The proof of this theorem is similar to that of Theorem 7, which is more interesting and is presented later, and is omitted.

Many existing multicast schemes can be shown to be special cases of our Construction 1 and their security can be readily predicted by Theorem 6. The Revocation method 1 in [93] and the group key distribution scheme in [5] are just Construction 1 instantiated with a special use of threshold ElGamal [3]. The basic scheme in [121], the "public key (multicast) encryption" from [93] and the CPA secure scheme from [44] can all be shown to be Construction 1 with a standard threshold ElGamal cryptosystem. These schemes are shown to be secure against chosen plaintext attacks in their individual papers. The same conclusion can be reached immediately through Theorem 6.

Theorem 6 also provides guidelines for constructing *new* multicast encryption schemes

[3]Their scheme uses this construction not to encrypt any useful messages. Instead, it is basically a distributed Diffie-Hellman key exchange which is equivalent to producing an ElGamal encryption of an arbitrary message (which is ignored) and allowing any member with proper keys to derive from the ciphertext, and partial decryptions, a secret key that can be used to encrypt actual data.

with guaranteed security. For example, some threshold schemes are known to be CCA2 secure (e.g. [116, 20, 1, 70] and the IND-CCA2 threshold ElGamal in [54]) and a multicast encryption constructed via Construction 1 using one of these schemes is therefore guaranteed to be CCA2 secure too. In addition, all existing ATD-based multicast encryption schemes [93, 5, 121, 44, 76] are based on discrete logarithm. Theorem 6 provides security guarantee for constructing multicast encryption using any other assumptions. For example, [38] provides a threshold RSA scheme with CPA security. Such scheme can be used to construct a RSA-based CPA secure multicast cryptosystem. Another example of factorization-based scheme is the threshold version of Paillier cryptosystem [98] presented in [54]. [54] provides techniques to make this scheme IND-CCA2. A multicast cryptosystem with the same level of security based on Paillier cryptosystem can thus be constructed using Construction 1. All the above examples have never been proposed before. They are the natural products of Construction 1 and their security is guaranteed by Theorem 6.

6.4.3 Extension to Construction 1

Construction 1 provides a simple way to utilize a threshold scheme to construct multicast encryption and we have shown that the resulting scheme is as secure as the underlying threshold scheme. It is basically an "encrypt-then-decrypt-t-times" scheme. It can be improved both in efficiency and security with simple extension.

In Construction 1, the encryptor has access to what are equivalent to t decryption shares in $\mathcal{TD}$ which are not available to an encryptor in the underlying threshold scheme. This gives her a chance to "protect" these shares and, as a result, the resulting multicast encryption can be made more secure than $\mathcal{TD}$. This can be seen as an extension of Construction 1:

Construction 1e (ME1e) *Same as Construction 1 except for the following:*

- *The encryption* E *produces ciphertext as* $\psi = (c, \{(j, \mathsf{D}_{\mathsf{TD}}(SK_j, c)) : j \in T' \cup R\}, \upsilon)$ *where* $c = \mathsf{E}_{\mathsf{TD}}(PK, m)$ *and* $\upsilon = \mathsf{Tag}(c, \Sigma, I)$ *is a "tag" for the ciphertext.*

- *The decryption* D *first computes* $\mathsf{Valid}(\Gamma_i, \psi, I)$ *where* Valid *is a checking function* *outputting* 0*, or* 1*. If* Valid *outputs 0,* D *returns* $\perp$*. Otherwise it proceeds the same as Construction 1.*

This construction can be used to build a multicast scheme with higher security than the underlying threshold scheme. This is essentially what was done in [44] and [76]. The protection mechanism (i.e. Tag and Valid) depends on the threshold scheme and the security goal. In [44], the standard techniques of [35] (which attaches tags to the ciphertext so that the recipients with proper keys can verify its validity) was applied to protect the decryption shares and the security achieved is what [44] called gCCA2 (Generalized CCA) which is a variant, and weaker version, of CCA2. To achieve real CCA2 security, [44] used secure message authentication code (MAC) to make the verification tags non-malleable. And [76] essentially used a threshold version of M-CS [20].

6.4.4 Sharable Trapdoor Permutation-Based Construction

A whole class of public key cryptosystems are based on trapdoor permutations. Let $f_{PK} : \{0,1\}^k \rightarrow \{0,1\}^k$ be a k-bit to k-bit trapdoor (one-way) permutation with inverse f_{SK}^{-1}, defined by the public-private key pair (PK, SK). A public key cryptosystem $\mathcal{E}^{f,g,h}$ encrypts a message m as $\mathsf{E}(m) = h(f_{PK}(g(m)))$ where g and h are probabilistic, invertible functions that specify pre- and post-encoding operations, respectively. Given a ciphertext c, the decryption algorithm D computes $u = h^{-1}(c), v = f_{SK}^{-1}(u)$ and $m = g^{-1}(c, u, v)$ [4]. Depending on the security, the decryption may involve computing $\mathsf{Valid}(c, u, v) \in \{0,1\}$ which is the verification of the encoding. The decryption returns $\perp$ if $\mathsf{Valid}(c, u, v) = 0$. We denote such cryptosystem as $\mathcal{E}^{f,g,h} = (\mathsf{KeyGen}, \mathsf{E}, \mathsf{D}, \mathsf{Valid})$ where KeyGen generates (PK, SK) on given security parameter 1^k. In the following, the keys will be dropped from the notations when there is no need to make them explicit.

Such cryptosystems are prevalent in practice. One example is the RSA Public Key Cryptography Standard # 1 [107], where $g(m)$ is essentially m padded with a string of

[4]Note that g and h are easily invertible and do not require trapdoors. Also note that both g and h are probabilistic and $g(m)$ maybe independent of m. In this case simply inverting v does not reveal m. However, these decryptions all have the following property: once the pre-image of the trapdoor permutation is recovered, it is easy to compute m. We simply use $g^{-1}(\cdot)$ to denote this process.

random non-zero bytes in the high-order bit positions and post-encoding is simply omitted. Other schemes make use of hash functions. Let $G : \{0,1\}^* \rightarrow \{0,1\}^\infty$ be a random number generator and $H : \{0,1\}^* \rightarrow \{0,1\}^{k_0}$ be a hash function where $l = k - k_0$ is the length of the message. In [8] Bellare and Rogaway proposed the scheme $\mathcal{E}_{BR}^{G}$ where $\mathsf{E}(m) = f(r) \parallel G(r) \oplus m$ with $r \leftarrow_R \{0,1\}^k$. [8] showed that it is semantically secure in the random oracle model. [8] also presented another scheme, denoted $\mathcal{E}_{BR}^{G,H}$, that is shown to be CCA2 secure, also in the random oracle model. In $\mathcal{E}_{BR}^{G,H}$, message m is encrypted as $\mathsf{E}(m) = (f(r), m \oplus G(r), H(r,m))$ where $r \leftarrow_R \{0,1\}^k$. Given a ciphertext (s,c,v), the decryption algorithm computes $r = f^{-1}(s), m = G(r) \oplus c$, and $v' = H(r,m)$. If $v' = v$, it outputs m, and $\perp$ otherwise.

Another popular scheme is the OAEP scheme introduced in [9]. In this scheme, to encrypt a message m of length l bits, one selects a random value $r \leftarrow_R \{0,1\}^{k_0}$ and computes $s = (m \parallel 0^{k_1}) \oplus G(r)$ and $t = r \oplus H(s)$ where $k_1 = k - l - k_0$. The ciphertext is $c = f(s,t)$. To decrypt a ciphertext c, the decryptor extracts (s,t) using the private key $(s,t) = f^{-1}(c)$ and computes $r = t \oplus H(s)$ and $M = s \oplus G(r)$. If $[M]_{k_1} = 0^{k_1}$, it returns $[M]^l$. Otherwise it returns $\perp$. In the above, $[M]_l$ (resp. $[M]^l$) denotes the l least (resp. most) significant bits of M.

In [9], Bellare and Rogaway proved that OAEP construction together with any trapdoor one-way permutation is IND-CCA1. OAEP was widely believed to achieve stronger security (i.e. IND-CCA2). But Shoup showed in [115] that it is unlikely such security proof exists, for any trapdoor permutation. However, he proved that, when instantiated with low-exponent RSA, OAEP was IND-CCA2. This result was extended to arbitrary exponent RSA in [55].

All these schemes provide practical public key cryptosystems with various security and efficiency. (The OAEP scheme provides optimal bit complexity in that the ciphertext size is only slightly greater than that of plaintext.) However, they do not have threshold implementations that retain the same security, especially at CCA2 level. As Shoup and Gennaro noted in [116], the difficulty in transforming a non-threshold CCA secure public key encryption scheme, $\mathcal{E}$, into a CCA secure threshold scheme is that $\mathcal{E}$'s security proof can rely in a critical way on the fact that the decryption algorithm makes the "validity test" before generating an output. In a distributed setting, this means the test can only be performed *after* the individual decryption shares are combined. A single decryption server is unable to carry out such test. Both $\mathcal{E}_{BR}^{G,H}$ [8] and OAEP can be easily shown to have this difficulty.

One way to address this difficulty is to introduce a validity test that is publicly checkable so that a decryptor can perform the check before carrying out the decryption. This was suggested in [82] and followed by systems such as [116] which used non-interactive zero-knowledge proofs of membership to construct such check which is costly.

An ATD-based multicast encryption scheme, on the other hand, does not suffer from this difficulty at all. This is because in such a scheme, the decryptor is presented with what are equivalent to t decryption shares in the underlying sharing scheme. She can proceed to combine these shares with the one produced using her private key and perform the simple validity test as in the original public key cryptosystem (not the expensive publicly checkable threshold version) *before* emitting any output. As we show in Theorem 7, this construction preserves the CCA security of the public key cryptosystem even though its threshold implementation does not.

Our new construction is based on sharable trapdoor functions.

Definition 4 (($t+1, n$)-Secure Sharing Scheme). *Let f be a trapdoor function with inverse f^{-1} defined by the public-private key pair (PK, SK). A sharing scheme $\mathcal{SS}^f = (\mathsf{S}, \eta)$ for f consists of two polynomial time algorithms:*

- S*: Given (PK, SK), a threshold t and an integer $n > t$, S generates $SK_1, \ldots,$ SK_n (in the same space as SK), called* shares *of SK.*
- η*: Given the public key PK, a set Λ of $t+1$ evaluations $f^{-1}_{SK_i}(u)$, for any u in the domain of f_{PK}, η computes $f^{-1}_{SK}(u)$.*

And $\mathcal{SS}^f$ is $(t+1, n)$-secure if for all $\{i_1, \ldots, i_j\} \subset U$ where $0 \leq j \leq t < n$, for all probabilistic polynomial time algorithm A, for all polynomial poly$(\cdot)$, for all k large enough

$$Pr[f_{PK}(u) = w : (SK_1, \ldots, SK_n) \leftarrow \mathsf{S}(PK, SK, t, n);$$
$$w \in_R \{0,1\}^k; u \leftarrow A(1^k, w, H, SK_{i_1}, \ldots SK_{i_j})] < 1/poly(k)$$

where H is the history tape of length polynomial in k containing all the partial evaluations the players generated so far.

And f is $(t+1,n)$-sharable if it has one $(t+1,n)$-secure sharing scheme.

This is essentially the same definition as $(t+1,n)$-secure function sharing primitive in [38]. [38] also showed how to implement such sharing with trapdoor permutations such as RSA. We show that using this primitive we can construct efficient multicast encryption schemes with high security.

Construction 2 (ME2). *Let $\mathcal{E}^{f,g,h} = (\mathsf{KeyGen_E}, \mathsf{E_E}, \mathsf{D_E}, \mathsf{Valid})$ be a public key cryptosystem based on $(t+1,n)$-sharable trapdoor permutation f with sharing scheme $\mathcal{SS}^f = (\mathsf{S}, \eta)$. Given a security parameter 1^k, a threshold t and the number of (initial) members n, a multicast encryption scheme $\mathcal{ME}_{C2}^{\mathcal{E}^f} = (\mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ can be constructed as follows:*

1. *Key Generation KeyGen: The center runs $\mathsf{KeyGen_E}$ with parameter 1^k, and obtains $(PK, SK) \leftarrow \mathsf{KeyGen_E}(1^k)$. It sets $I = PK$ and shares SK using the sharing algorithm S with parameter $(t+1, n+t)$ to obtain $\vec{SK} = \mathsf{S}(PK, SK, t, n+t)$. The encryption key is $\Sigma = \{(j, SK_j) : j \in T\}$ where $T = \{n+1, \ldots, n+t\}$. Σ is given to the center. Member i receives secret key $\Gamma_i = (i, SK_i)$. The master secret key is $\vec{\Gamma} = (\Gamma_1, \ldots, \Gamma_{n+t})$.*

2. *Encryption E: Given a set R of revoked members, and their secret keys, with $|R| \leq t$, a message m, the encryptor randomly selects a subset of T with $t - |R|$ elements, denoted T', and computes the ciphertext*

$$\psi = (c, \{(j, f_{SK_j}^{-1}(u)) : j \in T' \cup R\}) \tag{6.2}$$

where $c = \mathsf{E_E}(PK, m)$ and $u = h^{-1}(c)$.

3. *Decryption* D*: Given a secret key* Γ_i *and a ciphertext* ψ*, the ciphertext is first parsed into* $\psi = (c, \Lambda')$ *where* $\Lambda' = \{(j, f^{-1}_{SK_j}(u)) : j \in T' \cup R\}$*. The decryptor computes* $u = h^{-1}(c)$ *and* $v = \eta(u, \Lambda' \cup \{(i, f^{-1}_{SK_i}(u))\})$*. If all these steps are successful, it computes* $w = \mathsf{Valid}(c, u, v)$*. If* $w = 0$*, it returns* $\perp$*. Otherwise it returns* $m = g^{-1}(c, u, v)$*.*

Theorem 7. *Let* $\mu \in \{CPA, CCA2\}$*. If a public key cryptosystem* $\mathcal{E}^{f,g,h} = (\mathsf{KeyGen}_\mathsf{E}, \mathsf{E}_\mathsf{E}, \mathsf{D}_\mathsf{E}, \mathsf{Valid})$ *based on* $(t+1, n)$*-sharable trapdoor permutation* f *with sharing scheme* $\mathcal{SS}^f = (\mathsf{S}, \eta)$ *is secure against* μ *type attacks, then a multicast encryption scheme* $\mathcal{ME}^{\mathcal{E}}_{C2} = (\mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ *constructed using Construction 2 with threshold* t *and (initial) group size* n *is* t*-resilient against* μ*-type attacks.*

Proof. First note that it is trivial to verify that the scheme is correct – i.e., the decryption produces the correct plaintext given a valid ciphertext. We prove its security by showing that if $\mathcal{ME}^{\mathcal{E}}_{C2}$ is not t-resilient against μ-type attacks, neither is $\mathcal{E}^{f,g,h}$. Let $\mathcal{A}^{\mathcal{ME}}$ be a polynomial time adversary that wins the game ME with non-negligible advantage. We can construct another polynomial time adversary $\mathcal{A}^{\mathcal{E}}$ that breaks $\mathcal{E}^{f,g,h}$ with at least the same advantage. $\mathcal{A}^{\mathcal{E}}$ achieves this by simulating a game ME and running $\mathcal{A}^{\mathcal{ME}}$ to win.

$(PK, SK) \leftarrow \mathsf{KeyGen}_\mathsf{E}(1^k)$ is run and PK is given to $\mathcal{A}^{\mathcal{E}}$ while SK is kept secret from it. $\mathcal{A}^{\mathcal{E}}$ selects randomly t numbers $SK_1, \ldots, SK_t$ from the space of SK. $\mathcal{A}^{\mathcal{E}}$ starts Game ME and lets $\mathcal{A}^{\mathcal{ME}}$ select t members to corrupt. Without loss of generality, let $T = \{1, 2, \ldots, t\}$ be the indexes of the members $\mathcal{A}^{\mathcal{ME}}$ chooses to corrupt. $\mathcal{A}^{\mathcal{E}}$ simulates the key generation process in game ME and gives $\Sigma = ((1, SK_1), \ldots, (t, SK_t))$ as the corrupted keys and $I = PK$ as the public information to $\mathcal{A}^{\mathcal{ME}}$.

$\mathcal{A}^{\mathcal{E}}$ lets $\mathcal{A}^{\mathcal{ME}}$ run and simulates the rest of game ME as follows:

- Whenever $\mathcal{A}^{\mathcal{ME}}$ queries the encryption oracle with message m, $\mathcal{A}^{\mathcal{E}}$ returns ψ computed using Equation 6.2 with $T' \cup R$ replaced by T.

- $\mathcal{A}^{\mathcal{E}}$ chooses whatever $\mathcal{A}^{\mathcal{ME}}$ choose as the two test plaintexts m_0 and m_1. Whenever $\mathcal{A}^{\mathcal{ME}}$ makes a query to the encryption oracle with m_0 and m_1, $\mathcal{A}^{\mathcal{TD}}$ passes them to its own encryption oracle in its game attacking $\mathcal{E}^{f,g,h}$ (denoted game E). Let c' be the result returned by the encryption oracle in game E. $\mathcal{A}^{\mathcal{E}}$ computes and returns the following to $\mathcal{A}^{\mathcal{ME}}$:

$$\psi' = (c', \{(j, f^{-1}_{SK_j}(u)) : j \in T\}) \tag{6.3}$$

where $c = \mathsf{E}_{\mathsf{E}}(PK, m)$ and $u = h^{-1}(c')$. This corresponds to the target ciphertext in game ME.

- In the case of $\mu =$ CCA2, whenever $\mathcal{A}^{\mathcal{ME}}$ makes a query to one of the decryption oracles with ciphertext ψ, $\mathcal{A}^{\mathcal{E}}$ first parses ψ into a form as specified by Equation 6.2. Let $\{(j, u_j) : j \in T\}$ be the shares embedded in ψ. $\mathcal{A}^{\mathcal{E}}$ then verifies these shares by checking whether $u_j = f^{-1}_{SK_j}(u)$, where $u = h^{-1}(c)$, holds. If any of the tests fails it returns $\perp$ to $\mathcal{A}^{\mathcal{ME}}$. Otherwise it forwards c to its own decryption oracle and passes whatever the decryption oracle returns to $\mathcal{A}^{\mathcal{ME}}$.

$\mathcal{A}^{\mathcal{E}}$ stops when $\mathcal{A}^{\mathcal{ME}}$ stops and outputs whatever the latter does.

We need to show that $\mathcal{A}^{\mathcal{ME}}$ simulated by $\mathcal{A}^{\mathcal{E}}$ has all the information it would have in a real game ME and that its interaction with the simulated oracles is indistinguishable from that in a real game. First note that here, although the encryption key for $\mathcal{A}^{\mathcal{ME}}$, $SK_1, \ldots, SK_t$, are not actually generated by running S ($\mathcal{A}^{\mathcal{E}}$ does not have access to SK), they are just as good: the encryption key given to $\mathcal{A}^{\mathcal{ME}}$ is not distinguishable from that in a real game ME and does not affect its ability to win the game. This follows Lemma 1 from [38].

Second, $\mathcal{A}^{\mathcal{ME}}$ will receive $\perp$ on ciphertext ψ in the simulated game ME in one of the following two cases: (1) $\mathcal{A}^{\mathcal{E}}$'s decryption oracle returns $\perp$ on c; and (2) one of the tests on $u_j = f^{-1}_{SK_j}(u)$ fails. In the first case, $\mathcal{A}^{\mathcal{ME}}$ will receive $\perp$ in a real game ME, as specified by the decryption in Construction 2. In the second case $f^{-1}_{SK_j}(u)$, together with any partial evaluation of one of the decryption oracles in a real game ME, will combine to a u' that is not consistent with c and will fail Valid (otherwise it can be shown that either f is not

$(t+1, n)$-sharable or $\mathcal{E}^{f,g,h}$ is not IND-CCA2). Again $\mathcal{A}^{\mathcal{ME}}$ will receive $\perp$ in a real game ME.

And in all other cases $\mathcal{A}^{\mathcal{ME}}$ will receive the correct decryption in both real and the simulated game ME. So if $\mathcal{A}^{\mathcal{ME}}$ can win a real game, it can win the simulated one.

It is easy to verify that if $\mathcal{A}^{\mathcal{ME}}$ wins the simulated game ME, $\mathcal{A}^{\mathcal{E}}$ distinguishes the two target ciphertexts with at least the same advantage. This is because, by definition of Construction 2, if ψ' in Equation 6.3 is the encryption of $m_{b'}$ in $\mathcal{ME}$, c' must be the encryption of $m_{b'}$ in $\mathcal{E}$.

Finally $\mathcal{A}^{\mathcal{E}}$'s running time is polynomial in that of $\mathcal{A}^{\mathcal{ME}}$ which itself is a polynomial in k. So $\mathcal{A}^{\mathcal{E}}$'s running time is also polynomial in k.

□

This is very powerful result because securing threshold scheme is hard so it is not always possible to use Construction 1 to construct multicast cryptosystems with high security. Construction 2 and Theorem 7 offer a simple method to construct multicast schemes with guaranteed security using a whole class of existing primitives. For instance, both RSA-OAEP [55] and $\mathcal{E}_{BR}^{G,H}$ [8], which have been shown to be difficult to obtain threshold implementations with the same level of security, can be used to build multicast scheme with CCA2 security. This has never been achieved before.

Besides security, Construction 2 also enjoys higher efficiency than Construction 1, which directly uses a threshold scheme. Note that in a sharing scheme used by Construction 2, there is neither decryption share verification nor publicly checkable validity test on ciphertext, both of which are essential for a threshold scheme or a real function sharing application to achieve robustness (as in e.g. [57]) and CCA security. With Construction 2, both can be omitted and the encoding verification that is part of the public key cryptosystem used can achieve both goals.

6.4.5 From IND-CPA to IND-CCA: Generic Conversion

In Construction 2, the security of $\mathcal{ME}^{\mathcal{E}}_{C2}$ relies on that of $\mathcal{E}^{f,g,h}$. Combined with results from previous work, we show that $\mathcal{ME}^{\mathcal{E}}_{C2}$ can be IND-CCA even if $\mathcal{E}^{f,g,h}$ is only IND-CPA.

In [92] Naor and Yung presented a generic conversion from an IND-CPA public key cryptosystem to one secure against "lunch-time" attack (a.k.a. non-adaptive chosen ciphertext attack, CCA1). The conversion used a twin-encryption paradigm and non-interactive zero-knowledge proof (NIZKP) of language membership in the common random string setting to show the consistency of the ciphertext. Rackoff and Simon later [105] improved this construction to be secure against adaptive chosen ciphertext attack (CCA2). Their solution involves replacing one of the twin encryption keys with the *sender*'s public key and providing a NIZKP of knowledge of the plaintext. [54] also provided similar conversion, in the random oracle model, that also works directly with *threshold* cryptosystems. The NIZKPs used in [92, 105, 54] are all publicly verifiable thus can be readily used in a threshold setting.

Putting all these together, we have the following whose proof immediately follows the results of [54, 105, 92] and ours.

Corollary 1. *If a public key cryptosystem $\mathcal{E}^{f,g,h}$ based on $(t+1, n)$-sharable trapdoor permutation f with sharing scheme $\mathcal{SS}^f$ is secure against chosen plaintext attacks, then there exists a multicast encryption scheme $\mathcal{ME}$ by Construction 2 with threshold t and (initial) group size n that is t-resilient against chosen ciphertext attacks.*

SUMMARY. Figure 6.1 summarizes the possible conversions covered in this chapter between various primitives, including public key cryptosystem (PKC), threshold decryption scheme (TD) and multicast encryption (ME), at different security levels such as IND-CPA, IND-CCA (1 and 2). A solid arrow from A to B indicates "generic conversion", meaning that, under some reasonable assumptions, *any* A can be transformed into B. A dashed arrow, on the other hand, denotes "existential conversion", meaning that *some* A can be transformed into B. The conditions under which such conversions can succeed were stated in the literature. Some of the relevant ones covered in this chapter are labelled on the arrows.

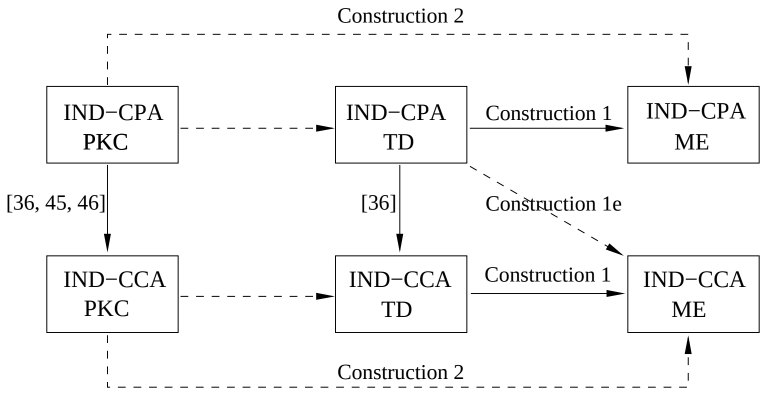

Figure 6.1. Conversions between various primitives. A solid arrow from A to B indicates "generic conversion", meaning that, under some reasonable assumptions, *any* A can be transformed into B. A dashed arrow, on the other hand, denotes "existential conversion", meaning that *some* A can be transformed into B.

6.5 Conclusion

In this chapter we have presented a general framework for constructing efficient multicast cryptosystems with provable security. Our constructions are based on asymmetric use of threshold schemes and we showed that a line of previous work on multicast encryption are all special cases of this general approach. We provided new methods for constructing multicast cryptosystems that achieve various levels of security (e.g., IND-CPA, IND-CCA2) from primitives with even weaker security. Using our scheme, each member only needs to store a key of constant length while both the encryption key size and the ciphertext length are $O(t)$ which is independent of the group size.

Chapter 7

Scalable Secure Bidirectional Group Communication

In this chapter we present a secure group communication scheme based on the IND-CCA2 multicast cryptosystem introduced earlier. Our scheme is bi-directional, supporting both one-to-many and many-to-one communications. Compared with existing solutions, the scheme achieves the following improvements: (1) It guarantees data confidentiality and authenticity in *both* directions; (2) It is the most scalable solution so far among all existing schemes achieving (1). The group member storage overhead is constant while both the center storage and rekeying communication complexity are independent of group size. (3) It can be made to achieve higher level of security and hide even the information about the group dynamics. We show that this protection is more effective and more efficient than existing solutions.

7.1 Motivation

A large number of existing and emerging network applications are based on *group communications* where one party, denoted the *center*, sends messages to a large number of authorized recipients and/or receives data from multiple sources. The one-to-many communication model is typically denoted *multicast* and we call the many-to-one pattern *aggregation*. Examples of group-oriented commercial applications include pay-per-view distribution of digital media, multi-player games, teleconferencing etc. Group communications can also be seen in many noncommercial and non-IP-based applications such as miliary communication, sensor network, etc.

Multicast offers a scalable solution to delivering the same messages to a group of receivers. On the Internet, multicast can be performed at both IP and application levels. For instance, at IP level, a multicast group is identified by a Class D IP address and any receivers can join or leave a multicast group by sending IGMP (Internet Group Management Protocol) [51] messages to their local router. Any sender can send message to a multicast group by addressing the message to the group address. The current IP Multicast is an "open" service in that it does not restrict delivery of data to a specified set of receivers. Access control in multicast must be achieved via other means. A typical solution is to encrypt data using symmetric-key encryption with a global traffic encryption key (TEK) that is known only to the multicast group members. The difficulty here is key management: the TEK may have to be changed whenever members join or leave the group. This is known as rekeying.

Compared to multicast, aggregation as a communication paradigm received less attention, especially in the context of IP networks, and was largely studied in the settings of other network communication such as sensor networks. However, we observe that, in a realistic application scenario, there are compelling needs for aggregation, even in "multicast" applications. For instance, in a pay-per-view broadcast system, a user needs to transmit messages to the center to request the service or confirm her agreement to pay for it.

Another area where many-to-one communication is becoming more and more important is network Intrusion Detection Systems (IDS). Most IDSes share a similar underlying structure: agents (or sensors, probes) reporting detections to a management system. This already falls into the aggregation communication paradigm. Furthermore, many present intrusion detection systems are known to have problems such as alert flooding, scalability, etc., and data aggregation is proposed to address these issues [40]. Data aggregation enables the system to group alerts together, analyze data in a broader context and distribute the load of handling alerts. A basic prerequisite for data aggregation is the support for many-to-one communication.

In this chapter we consider the security in *bidirectional* group communication (i.e., both multicast and aggregation) and present a scalable solution. Similar to unicast communication, there are two fundamental security properties in group communication: data *confidentiality* and *authenticity*. Confidentiality ensures that no parties other than the intended recipients should be able to access the message. Authenticity, on the other hand, is concerned with protecting the message against tamper and guaranteeing that it is indeed originated from the alleged sender. With existing technology, these two properties pose different challenges in the two communication modes we are considering. In multicast, authenticity is easy to achieve since there is only one sender and it is not much of a burden for each receiver to store the center's public key that can be used to verify its signature. In aggregation, on the other hand, it is confidentiality that can be obtained trivially, by using the center's public key to establish secure channels. However, authenticity is crucial in aggregation-based applications. For example, in an IDS, the system must ensure the authenticity of the data sent by each probe otherwise an attacker can inject false monitoring data and thwart the intrusion detection effort. Thus in this chapter we focus on achieving confidentiality in multicast and authenticity in aggregation.

Authenticating aggregation messages poses unique challenges and it is not easy to simply "patch" existing multicast schemes with aggregation security while still maintaining the same scalability. Let us denote the information that a user uses to authenticate her aggregation messages the user's *authentication key*. For the authentication to work, the authentication keys must not be global or public. The "obvious" solution of running a PRF on a user's ID and using the resulting random bits as authentication key simply does not work: since the IDs are public, anyone can produce the authentication key of any other users. Using the global multicast secret key has the same problem. Indeed, it appears that, without resorting to expensive digital signature, the center must share some pairwise secret with each members of the group for this purpose.

Efficiency wise, since in aggregation, the center is handling many data sources, the authentication method must be efficient otherwise the center will become a bottleneck. In addition, one must consider possible "hidden cost" when evaluating the feasibility of a scheme. Some schemes may feature smaller keys but they require the center to store a list of all active member IDs. The effective storage overhead is still linear in the size of the group. In a large scale system storing and accessing such a list can be too expensive.

In this chapter we present a scheme for bidirectional group communication that provides both confidentiality and authenticity in *both* directions. Our scheme is based on a new multicast framework recently proposed by Duan and Canny [47] that can be used to construct multicast cryptosystems provably secure against adaptive chosen ciphertext attacks (IND-CCA2) which is a very strong notion of security. We show that the key structure of the Duan-Canny (DC) multicast system provides a natural and efficient framework for sharing a pair-wise secret between the center and each member (to be elaborated in section 7.4). And this feature can be exploited to construct efficient group communication schemes that are secure in both directions. The contributions of our work include

- We define the communication paradigm of aggregation and formalize its security properties that are appropriate for most group communication applications.
- We introduce Alternating Bit DC (ABDC) to overcome the limitation of DC cryptosystem and allow the revocation of an arbitrary number of members.
- We present "in-place" update that provides backward confidentiality which is lacking in the original DC system.
- We extend DC construction's key arrangement to provide an efficient and scalable solution for authenticating aggregation messages. This is achieved while maintaining DC system's scalability. Our scheme uses symmetric key crypto for authenticating aggregation messages thus is much more efficient than digital signature-based solutions. In our scheme, the data authentication tag also serves as a group membership authentication which allows the center to verify the data authenticity and the sender's membership *without* having to keep the list of active member IDs online.

- We show that a scheme based on DC construction is affable to protecting group dynamics information (GDI) due to its property that the size of the ciphertext is independent of the group size. We present techniques that protect such information even against members of the group. To the best of our knowledge, this is the only effective GDI protection technique (to be elaborated in Section 7.6).

7.2 Preliminaries

We consider the situation where a single party, called *the center*, communicates, over insecure channels, with a group of n parties who are called *the members* of the group. The communication is assumed to be two way (i.e. both multicast and aggregation) and we require that security properties defined in Section 7.2.1 be guaranteed.

We assume the center is also trusted with managing the group membership. This is realistic in many applications (since oftentimes the center is the data distributor) and is in line with almost all existing multicast schemes such as [84, 128, 117, 126, 121, 44, 76]. We assume a computationally bounded adversary who is allowed to attack the system from both outside and *inside* the group. The insider's attack is modelled by allowing the adversary to corrupt and gain total control of up to t group members where t is a predefined threshold. The system deals with corrupted members by evicting them from the group.

We adopt the following notation in subsequent discussion. For both symmetric and asymmetric cryptosystems, let $\mathsf{E}_K(X)$ denote the encryption of X with key K and $\mathsf{D}_K(Y)$ be the decryption of Y with key K. We use $x \leftarrow y$ to denote that x is assigned the value of y. We write $x \leftarrow_R D$ to denote the process of selecting uniformly randomly from domain D an element and assigning it to x. We assume the process can be completed in constant time. We use $||$ to denote concatenation. For a bit b, we write $\bar{b}$ to denote its complement.

7.2.1 Security in Bidirectional Group Communication

The security notions in group communication are complicated by the fact that the group can be dynamic with members joining and leaving. Research in secure multicast has identified a number of security properties in dynamic group communication:

M.1 Non-group Confidentiality: users that were never part of the group should not have access to any multicast data sent to the group;

M.2 Forward Confidentiality: users deleted from the group at some time τ do not have access to any data after τ, unless they are authorized to join the group again;

M.3 Collusion Freedom: no subset of deleted users should be able to decrypt future group communication, even by sharing the keys they had before deletion;

M.4 Backward Confidentiality: a user added at time τ should not have access to any data before τ while the user was not part of the group.

Aggregation security can be defined as as the dual of multicast security. We introduce the following:

A.1 Non-group Authenticity: users that were never part of the group should not have the ability to forge messages that the center will accept as coming from one of its members;

A.2 Forward Authenticity: users deleted from the group at some time τ cannot create messages that the center accepts as originated from one of its members after τ, unless they are authorized to join the group again;

A.3 Collusion Freedom: no subset of t or less *active* members, nor any subset of deleted members, should be able to forge messages that the center accepts as originated from another member not in the colluding subset, even by sharing the their keys;

A.4 Backward Authenticity: a user added at time τ should not have the ability to forge messages that the center accepts as coming from a member who was in the group before τ.

Note that in A.3 we allow the collusion of (up to t) *active* members, not only deleted ones. This is different from the collusion freedom for multicast (M.3). In multicast, the primary concern is data confidentiality. Any active member by definition should have access to the multicast message so it does not make sense to protect against their collusion. M.3 is therefore actually expressed as a stronger version of forward confidentiality (M.2). We keep the definitions separate because they emphasize different aspects of the system's security (temporal and collusion-resilience, respectively) for which most previous multicast security works kept separate definitions.

The primary security goal for aggregation, on the other hand, is data authenticity which should be resilient against forgery and tamper even by active members. A.3 defines such attack by, in addition to deleted members, allowing a subset of t or less active members to collude. Note that A.4 is actually implied by A.3. We keep a separate definition to stress the temporal aspect of the security goal.

Also note that the definitions state that a scheme should be resilient against collusion of *any number* of deleted members. The threshold t only restricts the number of colliding *active* members. These properties provide basic security goals for many bidirectional group communication applications and will be used as guidelines for designing and evaluating our scheme.

In addition, some applications may demand even higher level of security and require that the information about the group dynamics such as group size and user join/departure rate be kept secret from an adversary. We discuss this issue in Section 7.6 and show that our scheme can be made to effectively protect such information.

7.3 Overview of DC Multicast Cryptosystem Construction

In this section we give an overview of Duan-Canny multicast cryptosystem construction. We focus on how DC multicast cryptosystem works and omit some non-relevant elements from the notation. For detailed definition and proof of security please see [47].

Fundamental to [47]'s constructions is a public-key cryptosystem with a sharable decryption. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ be such a cryptosystem. It consists of the following algorithms:

1. Key Generation KeyGen: a probabilistic polynomial-time algorithm that, on a given security parameter 1^k (expressed in unary), generates a private-public key pair (x, y).
2. Encryption E: a probabilistic polynomial-time algorithm that, on inputs y, the encryption key, and a string $m \in \{0,1\}^k$, produces as output $\psi \in \{0,1\}^*$ called the ciphertext.
3. Decryption D: a deterministic polynomial-time algorithm such that $\forall m \in \{0,1\}^k$, $\mathsf{D}_x(\mathsf{E}_y(m)) = m$. On all other inputs it outputs a special symbol $\perp$.

There also exists a share generation algorithm that, given (x, y), a threshold t and an integer $n > t$, generates $x_1, \ldots, x_n$ (in the same space as x), called *shares* of x such that any subset of at least $t+1$ shares uniquely determines the secret key x while any subset with less than $t+1$ shares has no information about x. In addition, D should be "sharable": given a ciphertext ψ and a share x_i of the secret key, D produces a share of the decryption $\psi_i = (i, \mathsf{D}_{x_i}(\psi))$ such that given a set Λ of the decryption shares with $|\Lambda| > t$, a "share combination" algorithm can produce the correct decryption $\eta(\Lambda, y) = \mathsf{D}_x(\psi)$.

DC encryption works as follows. The center generates a public/private key pair (e.g., ElGamal key pair), publishes the public key y and secret-shares the private key x using $(t+1, n+t)$-threshold scheme among all group members and the center. Let $x_1, x_2, \ldots, x_{n+t}$ be the shares of x. Each member i gets x_i and the center is given $\{(j, x_j) | j \in T\}$ where $T = \{n+1, n+2, \ldots, n+t\}$. To multicast the message $m \in \mathcal{M}$, the center encrypts it using y. Let $c = \mathsf{E}_y(m)$. It then decrypts c t times using its shares of x and produces $\bar{m} = \{(j, m_j = \mathsf{D}_{x_j}(c)) | j \in T\}$. The ciphertext is $\psi = (c, \bar{m}, T)$. The center multicasts ψ to the group. To decrypt the message, member i first decrypts c using x_i, her share of the private key, and then combines all the partial decryptions she has to obtain

$$m = \eta(\bar{m} \cup \{(i, \mathsf{D}_{x_i}(c))\}, y)$$

DC encryption can revoke up to t members. Let R be the indices of the members whose capability to decrypt the message should be revoked. The center construct a "blacklist" $\bar{T}$ as $\bar{T} = R \cup T'$ where T' is a random subset of T with $t - |R|$ elements. The center then use $\bar{T}$ in place of T in the encryption. The members whose indices are in R will not be able to decrypt the message.

[47] proved that their scheme is secure against adaptive chosen ciphertext attack (IND-CCA2) when $\mathcal{E}$ is IND-CCA2. This security is defined against up to t colluding members. In the following, to take advantage of this guaranteed high security, we will assume an IND-CCA2 DC cryptosystem.

7.4 Scalable Bidirectional Group Communication

To make it concrete, we describe our group communication scheme using a DC multicast cryptosystem instantiated with Shoup and Gennaro's `TDH2` threshold scheme in [116]. This instantiation was based on the decisional Diffie-Hellman (DDH) problem and proven secure against adaptive chosen ciphertext attack. For detailed proof and description of the scheme please see [116] and [47]. For the purpose of this chapter, it suffices to know that the encryption is essentially ElGamal augmented with non-interactive zero-knowledge proof of discrete logarithm (to thwart chosen ciphertext attack, which we omit). We defer the description of the scheme until after we introduce some improvements over the original DC cryptosystem.

7.4.1 Extension for Aggregation

Although DC cryptosystem is for multicast encryption, its key structure has a very nice property that can be used to efficiently authenticate *aggregate* messages. Recall that the center is trusted with both distributing messages and managing the group. In this case the center can be given $t+1$ shares of x. Let $T^+ = T \cup \{n+t+1\}$ and $\Gamma = \{(i, x_i) | i \in T^+\}$ are given to the center. Each member i attaches to an aggregation message a secure message authentication code (MAC) computed with a key derived from x_i. The center first recover

x_i using Γ and then tests the validity of the MAC. This uses symmetric crypto and is much more efficient than digital signature. This is essentially sharing a distinct secret key between the center and each member, *without* having to require the center to store all of them. The center stores only $t + 1$ keys instead of n and recovers the member keys as needed. Theoretically, using only these $t + 1$ keys, the center is able to authenticate an *arbitrary* number of members, limited only by the system's assumption on the proportion of members that can be simultaneously corrupted. This provides a scalable solution for large groups.

7.4.2 Alternating Bit DC

DC cryptosystem allows for the revocation of up to t members. This count is accumulated from the start of the system. If the total number of members that need to be revoked during the life time of this group communication should exceed t, their scheme must be "refreshed" with new public key and private key shares. This process cannot be carried out using multicast and must resort to the inefficient pairwise unicast communication. Yet the need to evict more than t members is quite compelling in many applications and this limitation may force the communication to pause for the refreshing.

To overcome this limitation, we devised an improved scheme, denoted Alternating Bit DC (ABDC), that moves the refreshing process off-line and allows for uninterrupted communication. ABDC enables the revocation of up to t members for each rekeying operation and allows arbitrary number of members, limited by the size of the group, to be revoked during the lifetime of the communication.

ABDC alternates between two Duan-Canny schemes, denoted DC_0 and DC_1. The system maintains a bit b that specifies which is currently being used and toggles the bit when one scheme is approaching its limit of t revocations. This information is included in the rekeying message and members will respond by switching to the other DC scheme. The previous DC scheme is then refreshed off-line. The transition is illustrated in Figure 7.1 and the process and messages will be clear in Section 7.4.6.

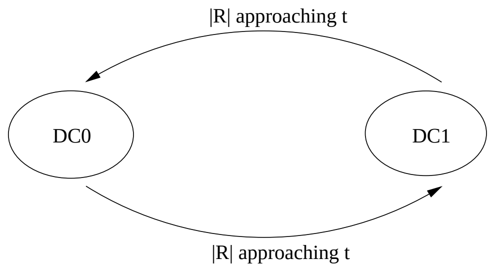

Figure 7.1. Alternating Bit DC

7.4.3 Initialization

This stage is carried out when the communication group is first being formed. Let n be the current size of the group. Let p and q be two large primes selected by the key generation algorithm as defined by the DC cryptosystem on a security parameter 1^k such that $q|p-1$. We use G_q to denote the unique subgroup of $\mathbb{Z}_p^*$ of order q and let g, $\bar{g}$ be a generators of G_q. DDH is assumed to hold in G_q

For $b = 0, 1$, the center performs the following:

1. It generates $t+1$ uniformly random numbers $f_{b0}, f_{b1}, \ldots, f_{bt} \in \mathbb{Z}_q$ and constructs a degree t polynomial $f_b(x) = \sum_{j=0}^{t} f_{bj} x^j$.
2. It computes $x_{bi} = f_b(i) \bmod q$ and $h_{bi} = g^{x_{bi}} \bmod p$ for $i = 1, 2, \ldots, n+t+1$. Let $x_b = f_{b0} = f_b(0)$ and $h_b = h_{b0} = g^{x_b} \bmod p$.
3. It sends x_{bi} to member i via secure channel and stores $\Gamma_b = \{(j, x_{bj}) | j \in T^+\}$. h_b is made public.
4. It sets $R \leftarrow \emptyset$ where R is the set of indexes of the members whose memberships have been revoked.

The public key y_b for DC$_b$ consists of a description of G and $(g, \bar{g}, h_b)$. The private key for each member i is (y_b, i, x_{bi}). The private key for the center is (y_b, Γ_b) [1].

[1] We omit the details about the public verification key which consists of the tuple $(y, h_1, h_2, \ldots, h_n)$ and is for the purpose of verifying the validity of partial decryptions. They are not necessary here since the partial decryptions are produced by the center who is the data source.

When the center is sending the keys to the members, it includes TEK_0 and b as well, also via the secure unicast channels. TEK_0 is the current traffic encryption key (TEK) and $b \in \{0, 1\}$ specifies the DC scheme currently being used.

7.4.4 Multicast

The center encrypts data using TEK_0, the current traffic encryption key. To guarantee data authenticity, it attaches its signature on the data. Each member stores the center's public key that can be used to verify the signature. TEK_0 and public key signature are enough to guarantee confidentiality and authenticity of the multicast data. The only overhead on a member is the the center's public key [2].

7.4.5 Aggregation

As mentioned in Section 7.4.1, we can use existing key structure in the DC cryptosystem to authenticate aggregate messages. The authentication is via a Message Authentication Code (MAC) with a key derived from her share of the secret key x_{bi}. The derivation can be done via a cryptographically secure pseudorandom number generator (PRNG) with x_{bi} (and maybe some other randomness) as seed. In practice, it can be as simple as using a cryptographically secure hash function. Let H be such a hash function. Member i first generates a random initialization variable IV of sufficient length (e.g. $IV \leftarrow_R \mathbb{Z}_q$) and the authentication key ak_i is computed as $ak_i = H(x_{bi}||IV)$. The randomness-like property of the hash makes the output look independent of the input, avoiding the possible vulnerability of using the same key in multiple cryptographic operations. This is a common practice found in many key manage schemes such as [102, 114, 101] etc., and the SSL protocol. The center, who is in possess of the polynomial that generates x_{bi}, can recover x_{bi} and generate ak_i to verify the MAC. The detailed protocol is described in figure 7.2.

[2] There are schemes for authenticating multicast data that require minimal buffering on the sender side and resist packet loss (e.g., [100, 106, 127, 59]). Some achieved great efficiency by making additional assumptions such as weak time synchronization between sender and receivers [100, 101]. However, as [12] showed, the general multicast message authentication problem is equivalent to digital signatures. We take the general approach here and leave the rest as options for applications.

Procedure Member i Aggregation

Input: Aggregation message m, member key x_{bi}.

1. Generate $IV \leftarrow_R \mathbb{Z}_q$ and $ak_i = H(x_{bi}||IV)$.
2. Send to the center $M_i = (i||m||IV||\mathsf{MAC}_{ak_i}(i||m||IV))$.

Procedure Center Verify

Input: Revoked set R, center key Γ, message M_i

1. Parse M_i into $M_i = (i||m||IV||tag)$.
2. If $i \in R$, reject the message.
3. Recover x_{bi} via Lagrange interpolation:
   $x_{bi} = \sum_{j \in T^+} L_j(i) x_{bj}$ where $L_j(x) = \frac{\prod_{k \in T^+, k \neq j}(x-k)}{\prod_{k \in T^+, k \neq j}(j-k)}$.
4. Compute the authentication key by $ak_i = H(x_{bi}||IV)$.
5. If $tag = \mathsf{MAC}_{ak_i}(i||m||IV)$, accept the message. Otherwise reject it.

Figure 7.2. Procedures for (a) member aggregation, and (2) center verification.

7.4.6 Rekeying

The rekeying protocol is summarized in figure 7.3. Essentially it uses DC system to multicast a new TEK such that the revoked members cannot access. This is achieved via DC system's built-in revocation capability. We added the "increment by δ" step which essentially achieves "in-place update" of DC$_b$. It shifts the DC scheme by a random (but secret) amount and a newly admitted member has no information about past DC scheme even if it has not been completely refreshed. As will be shown in Section 7.5 this feature guarantees backward security which is lacking in the original DC cryptosystem. And this is carried out via multicast and is very efficient.

7.4.7 Refresh

This procedure is used to refresh DC$_{\bar{b}}$ when it has approached its limit on the number of revocations. The center generates a fresh DC cryptosystem as it does in the Initial-

```
Procedure Center Rekey
Input: Revoked set $R$, center key $\Gamma$.
1. Select randomly $T' \subseteq T$ such that $|T'| = t - |R|$ and set $\bar{T} \leftarrow T' \cup R$.
2. Generate a new TEK, denoted $TEK_1$, and compute
   $c = \mathsf{E}_{y_b}(TEK_1||\delta||s)$ where $\delta \leftarrow_R \mathbb{Z}_q$ and $s = 1$ if $|R|$ is close
   to $t$ and 0 otherwise.
3. Multicast the following rekey message:
      $M = (c, \{(j, \mathsf{D}_{x_j}(c))|j \in \bar{T}\}, \mathsf{E}_{TEK_0}(\bar{T}))$
4. If $s = 1$, set $b \leftarrow \bar{b}$ and carry out Refresh as a
   separate procedure asynchronously. Also reset $R$ to $\emptyset$.
5. Update the public keys by $h_b \leftarrow h_b g^{\delta} \bmod p$, and it shares of the
   private key by $x_{bj} \leftarrow x_{bj} + \delta \bmod q, \forall j \in R \cup T^{+}$.
```

```
Procedure Member $i$ Rekey
Input: Rekey message $M$, member key $x_{bi}$.
1. Decrypt $M$ using $x_{bi}$ and obtain $TEK_1$.
2. Set $TEK_0 \leftarrow TEK_1$ to be the current group key.
3. If $s = 1$, set $b \leftarrow \bar{b}$.
4. Increment $x_{bi}$ by $\delta$, i.e., set $x_{bi} \leftarrow x_{bi} + \delta \bmod q$, and update
   the public key by $h_b \leftarrow h_b g^{\delta} \bmod p$.
```

Figure 7.3. Procedures for (a) center rekeying, and (b) member rekeying.

ization stage and sends the new keys to each member via secure unicast channel [3]. This process is "off-line" compared to multicast and aggregation and can be performed anytime before it needs to run again. All operations in the group communication, including data communication and rekeying etc., can proceed uninterrupted using DC$_b$.

[3]These secure unicast channels can be constructed, for example, by letting the members initiate a key agreement protocol using the aggregation mode as described in Section 7.4.5. The center does not need to store all public keys of the members.

7.4.8 User Join

When a user is allowed to join the communication, the center selects an unused index i and sends her all information corresponding to her index as specified by the key generation procedure. At this point we distinguish two types applications. The first allows the newly admitted members to access previous multicast data during the same session (i.e., the data encrypted with the TEK_0). In this case the center simply sends TEK_0 along with the key information to the new member via secure unicast channel. No rekeying message is necessary. The second type requires a strict backward security and prohibits the new member from accessing any data prior to her admission. For this type of applications the rekey procedure is invoked before the center sends the key information to the new member. We will show later that this scheme guarantees the strict backward security.

7.4.9 User Departure

To remove a member i from the group, the center recovers her share of the private key x_{bi} and adds i to R. It then invoke the Rekey procedure.

7.5 Analysis and Evaluation

In this section we evaluate our protocol in terms of security and efficiency. Since our protocol incurs constant cost in terms of all resources (storage, communication and computation) for each rekey operation, the evaluation here will be mostly analytical. The actual cost of any application is dependent solely on the group dynamics.

7.5.1 Security

The security of our protocol is summarized in the following theorem.

Theorem 8. *Assuming the security of the DC cryptosystem, the MAC algorithm and the* *symmetric key encryption scheme used for data encryption, the protocol described above satisfies the 8 properties specified in Section 7.2.1.*

Proof. Assuming the security of the DC cryptosystem (which has been proven in [47]), the MAC algorithm and the symmetric key encryption used to encrypt the data, for an adversary to successfully "attack" our scheme, she must obtain enough information to recover the appropriate key(s), where "attack" means gaining access to data in multicast and forging messages in aggregation. We show that for any attempted attack on any of the properties, this is impossible. Let ATKi denote the attack on property Mi and Ai where $i = 1, 2, 3, 4$.

ATK1 We show that an outsider cannot obtain any member's key by observing network traffic. There are three types of traffic in our protocol: key distribution traffic, data traffic, and rekey traffic. The first is via secure unicast channels and the second is encrypted with the current TEK. Both, by definition, provides confidentiality to those without proper keys. The rekey traffic is characterized by messages as specified in figure 7.3. This is essentially the ciphertext as defined in DC cryptosystem [47], with the blacklist encrypted with the current TEK and is shown to be secure against adaptive chosen ciphertext attacks. Therefore the attacker cannot get the keys to mount attack on M.1 or A.1.

ATK2 When a member i is removed from the group, according to our protocol, two events will happen: (1) the rekey procedure will be invoked and (2) her index i will appear, and remain, in the center's revocation list R until next time the system switches to the other DC scheme when R will be reset to $\emptyset$. The rekey procedure essentially utilizes the revocation capability of DC cryptosystem and guarantees that i will not be able to obtain new traffic encryption keys as long as i remains in R or a fresh DC scheme is used (to which i does not have access). Note that in our protocol, the rekey message that causes the transition to a new DC scheme will also cause all remaining members to update the DC scheme that will be used for next rekey operation (i.e., the increment of private keys by δ in the last step on both the center and members. See Section 7.4.6). Since the revoked member does not have access to this rekey message, she cannot obtain δ thus her information about the new DC scheme is obsolete and cannot enable her to decrypt future rekey messages even though R is reset to $\emptyset$. This guarantees forward confidentiality (M.2).

As for forward authenticity (A.2), note that when verifying the MAC, the center checks the sender's index against R. When i is in R, her MAC will always be rejected. After R is reset to $\emptyset$, both her private keys x_{0i} and x_{1i} are no longer valid any more because x_{bj} has been updated by δ and $x_{\bar{b}j}$ has been completely refreshed in an off-line process for all other legitimate members. Therefore the center will reject all her MACs keyed using her old keys. This guarantees forward authenticity (A.2).

ATK3 Attack on collusion freedom is modelled by allowing the adversary to control up to t members. Confidentiality in multicast (i.e., M.3) comes directly from the security of DC cryptosystem [47]. Aggregation authenticity (A.3) in face of up to t colluding members is guaranteed due to similar property of the secret sharing scheme [112].

ATK4 When a user is added to the group and assigned the index i, the center first invokes the rekey procedure. The user is then given her keys in both DC$_0$ and DC$_1$ and the current TEK via secure unicast channel. This allows her to access current multicast data and future rekey information. Note that the latest rekey procedure prior to her admission updates the current DC cryptosystem for existing members. This means the keys she received via the unicast channel will not enable her to decrypt previous rekey messages which is the only means to distribute TEKs. Therefore she cannot have access to data multicasted before she is admitted (M.4).

A.4 is implied by A.3. □

7.5.2 Efficiency

The biggest advantage of our scheme is the feature that each member only needs to store 1 constant-sized key which is truly independent of the group size. This can be significant for systems where the members are limited in storage. The center is required to keep a key of length $O(t)$ to be able to authenticate all n members which is much less than the

$O(n)$ storage overhead required for maintaining data authenticity in aggregation using other schemes. Note that the aggregation MAC can also service as a *membership* authentication. This is because, due to the security of the secret sharing scheme, the MAC algorithm and the cryptographic hash, only legitimate member with x_{bi} can produce such tag. And in our scheme, verifying the MAC only requires the use of center's own key (in contrast, in a digital signature-based scheme such as identity-based signature schemes, the center must retrieve the sender's public key to authenticate the message and verifying membership requires consulting an active member list). This means that, unlike other schemes, there is no "hidden cost" in our scheme: the information to authenticate aggregation messages (and memberships) is all contained in the center's key and the center does not even have to store a list of user IDs online.

The communication complexity for each rekey message is $O(t)$, also independent of the the group size. This is a useful feature not only from a scalability point of view, but also for the purposes of hiding group dynamics information, as will be explained in Section 7.6.

Rekeying involves public key encryption/decryption which are relatively inefficient to compute compared to symmetric key schemes. When instantiated with Shoup and Gennaro's `TDH2` threshold scheme [116], both encryption and decryption involve $O(t + 1)$ exponentiations. For encryption, since the bases g, $\bar{g}$ are fixed (per public key), one can pre-compute a table to speedup the operations significantly. For decryption, what is really needed is the *product* of the exponentials. The cost can be reduced considerably by using techniques such as simultaneous multiple exponentiations (Chapter 14.6.1 in [88]). Moreover, the computation is highly parallelizable. Each exponentiation is independent of another and can be carried out concurrently. This is true for both the center and the members.

Another advantage of our scheme is that authenticating aggregation messages uses symmetric key crypto. The efficiency obtained is critical for large-scaled systems supporting a large number of members who generate data at a high rate (e.g. a large IDS). Using the latest Crypto++ benchmark (http://www.eskimo.com/∼weidai/benchmarks.html), HMAC, a popular MAC algorithm with proven security [7], features a throughput of 216.674 MB/s. This is superior to any public key crypto-based schemes.

7.6 Hiding Group Dynamics Information

In an INFOCOM '04 paper [120], Sun and Liu first raised the issue of protecting group dynamics information (GDI) in multicast. They showed in [120] that in many schemes, the key management messages can disclose information about the dynamics of group membership such as group size and the user join/departure rate. This information can be used by an adversary or competitor in a way that is detrimental to the multicast group. [120] outlined two techniques an attacker can use to obtain GDI. The first technique applies to schemes where the attacker can distinguish rekey messages caused by user join from those resulted from user departure: he can conclude, with high certainty, that a user joins the group when he sees a join rekey message and a user leaves the group when a departure rekey message is observed. The second technique estimates the group size from the rekey message size when the two are related. The first attack is typically mounted by a group member who can decrypt the rekey messages and the second one can be launched by an outsider observing the traffic. [120] showed that most tree-based centralized key management schemes (e.g. [90, 125, 19, 128, 124]) are vulnerable to both attacks. Variants of these attacks are also effective on some other schemes. Please see [120] for details.

As observed in [120], the fundamental cause of their vulnerability is that most key management schemes allow an inside attacker to separate the rekey messages for user join and those for user departure and/or produce rekey message whose size is dependent on the group size. [120] proposed using batch rekeying and phantom users, who join and leave the group artificially, to augment existing schemes to conceal GDI. However, batch rekeying is only possible when it is acceptable to delay updating the session key in face of group membership change. Phantom users causes much overhead in terms of communication and computation. Neither technique is free and completely satisfactory. In particular, the effectiveness of phantom users is totally dependent on the artificial group size (which is the total number of real and phantom users), denoted N_0 as in [120]. The approach proposed in [120] is "all-or-nothing" in that if the actual group size, N, never exceeds N_0, it is perfectly hidden. Should N becomes greater than N_0 for some time, the adversary can obtain accurate estimate on N for the entire duration when $N > N_0$. To actually conceal group size, a large number of phantom users must be used to ensure N_0 is greater than the maximum number of users the group can ever accommodate. This is a serious problem for highly dynamic groups and the approach does not provide a really scalable solution. Yet there seems no better way out for schemes where the rekey message size is dependent on group size. This is one of their inherent limitations.

If we only consider the leakage by multicast and key management messages, as is done in [120] [4], we can show that our scheme has some nice properties that make it more affable to hiding GDI. First of all, it is completely immune to attacks on GDI mounted by *outsiders*. This is due to the following two features: (1) the only information in the rekey message that reveals GDI is the blacklist $\bar{T}$ and it is encrypted with current session key; and (2) the size of rekeying messages is independent of group size. Even if an outsider can observe the rekey messages, he won't be able to differentiate user join events from user departure, nor can he obtain any information about the size of the group.

As for the insider's attack, we first observe that, while revealing GDI to *outsider* is hardly ever acceptable, in some applications, it is actually beneficial to disclose group dynamics information to legitimate nodes of the system. For example, in some network, knowing the compromised nodes allows the good ones to avoid routing messages towards them. In this case, our scheme does not need to change at all. The blacklist $\bar{T}$ that is part of the rekey message provides a natural vessel for conveying the information about who have been evicted to legitimate members.

If GDI has to be hidden even from legitimate members, we introduce the following modifications. Note that our scheme as described so far already guarantees that the message

[4] Other leakages are possible, e.g. via the initialization. They are common to all schemes and fixing them is orthogonal to securing the multicast messages. Besides monitoring all other traffic is substantially harder than listening to the multicast channel, which can easily be done by joining the group. And even if such monitoring is possible, it is often possible to "bury" group dynamics-related non-multicast messages in other traffic. Therefore we take a similar approach as [120] and focus on multicast management messages.

size is independent of the group size even to an insider. All we need to fix is to "mix" the events of user join with departure.

1. We use batch rekeying and invoke the Rekey procedure only when the system has accumulated some number of join/departure users. [5]

2. In initialization, the center selects a large number N_0 greater than the upper bound of the group size. $\mathbb{Z}_{N_0}$ will be the space where user indices are drawn. For $s = 0, 1$, each of the initial set of users will be assigned a unique but random index from $\mathbb{Z}_{N_0}$. Note that each member gets independent indices for DC_0 and DC_1.

3. $\mathbb{Z}_{N_0}$ for each DC scheme is divided into 3 disjoint sets. For DC_s, where $s = 0, 1$, A_s is the set of indices currently assigned to active members. R_s is the set of indices belonging to the users whose membership have been revoked using the current DC scheme. MP_s is the "mixing pool" whose purpose will be clear later. During Initialization A_0 and A_1 are initialized according to the index generation process, R_0 and R_1 are all set to $\emptyset$.

4. The blacklist $\bar{T}$ for each rekey message is constructed as usual by including R_b and subset of MP_b [6]. Each rekey message will set $s = 1$ which causes the system to switch to the other DC scheme.

5. When switching to a new DC scheme, the center sets $b \leftarrow \bar{b}$ and $R_b \leftarrow \emptyset$.

6. When a user joins the group, for $s = 0, 1$, the center randomly selects a number i_s from MP_s as her index for DC_s and generates the key information accordingly. It then sets $MP_s \leftarrow MP_s \setminus \{i_s\}$, $A_s \leftarrow A_s \cup \{i_s\}$.

7. When a member, with indices i_b and $i_{\bar{b}}$ for DC_b and $\mathrm{DC}_{\bar{b}}$ respectively, leaves the group, the center sets $A_b = A_b \setminus \{i_b\}$, $A_{\bar{b}} = A_{\bar{b}} \setminus \{i_{\bar{b}}\}$, $R_b = R_b \cup \{i_b\}$, $MP_{\bar{b}} = MP_{\bar{b}} \cup \{i_{\bar{b}}\}$.

Theorem 9. *The modified protocol hides GDI from an insider.*

[5] The consequence of this modification is that the forward and backward security properties are only guaranteed at a coarser granularity, determined by the intervals between the Rekey operations.

[6] Note that the center does not maintain set T anymore.

Proof. (Sketch) First because of the batch rekeying, user join and departure are not distinguishable from rekey messages. Second, the system switches to a fresh DC scheme after each rekey message and the set R, whose elements will appear in $\bar{T}$, will be completely independent for the next rekey operation. An adversary cannot build her knowledge of revoked members, or the new members by observing multiple rekey messages. $\square$

With these modifications, the costs for member storage, member computation, rekey communication all remain the same. The only added overhead is for the center to maintain these sets. We can use the following to mitigate the cost.

First, if the center has enough storage capacity, these sets can be maintained as linked lists to minimize computation cost. Although the storage is now $O(n)$, the computation remains $O(t)$.

If, on the other hand, the center is limited on storage, these sets can be maintained as follows. For each DC scheme, the center maintains 2 N_0-bit numbers, N_A and N_R. The membership of an index i with respect to the 3 sets is encoded by the two bits $N_A[i]$ and $N_R[i]$ where $X[i]$ means the ith bit of X in a binary representation. We define the following encoding: $i \in A$ if $N_A[i] = 1$, $i \in R$ if $N_R[i] = 1$ and $i \in MP$ if $N_A[i] = N_R[i] = 0$. Note when a user leaves the group, her indices are all known and updating the sets amounts to simply flipping bits at known locations. The most expensive computation is when a user joins the group and the center needs to find an unused index in MP for her. For this purpose the center generates a random number i in the range of 1 to N_0. If $N_A[i] = N_R[i] = 0$, then i is assigned to the user and the sets are updated accordingly. Otherwise the center tries another random number. The probability that the center cannot find an unused index in MP after k tries is $(\frac{n}{N_0})^k$. Assuming N_0 is at least twice of n, with high probability, the center can find an unused index in MP with just a few tries. The computation can still be bounded by $O(t)$.

In any case the center storage complexity becomes $O(n)$. However, we point out that this is the same for all other secure group communication schemes if they are used for bidirectional communications even without any consideration for protecting GDI. And in our 2nd scheme, the constant behind the O is much smaller: the center in our scheme as just described only needs to store 4 N_0-bit numbers while in all other schemes must maintain at least n l-bit numbers where l is the length of the key used.

7.7 Related Work

In early secure multicast schemes (e.g., Group Key Management Protocol (GKMP) [66]), the center shares a pairwise key with each group member and distributes group keys to group members on a one-to-one basis. For obvious reasons this cannot scale to large groups. Some work has been done to improve the scalability of such schemes. Among the efficient solutions, the Logical Key Hierarchy (LKH) (or Key Graph) was independently discovered in [125] and [128] and has been an inspiration for many subsequent works [19, 26, 129, 81, 111, 131]. In these schemes, individual and auxiliary keys are organized into a hierarchy and each group member is assigned to a leaf and holds all the keys from its leaf to the root. The root key is shared by all group members and used as the TEK. New TEK is distributed by encrypting it with keys that deleted members do not have. So far $O(\log n)$ seems to be the best storage (for both center and members) and communication complexity the LKH-based schemes achieved.

Asymmetric key cryptosystems are also used for multicast key management. This includes the work in cryptography such as *traitor tracing*, a concept introduced by Chor, Fiat and Naor [27], and *broadcast encryption*, initiated by Fiat and Naor [52], and a recent multicast encryption framework formalized by Duan and Canny [47]. These schemes can be used to distribute the new TEK in the rekeying operation.

All these schemes focus on achieving private communication in the multicast direction. Little attention was paid to aggregation. As we mentioned before, the global TEK is not enough for data authentication in aggregation and these schemes do not have other provisions for such purpose.

There are some techniques that can be used for authenticating aggregation messages and they appear to enjoy reasonable scalability. For example, identity-based signature (IBS)

[113, 25] allows one to verify another party's signature using the signer's identity information (e.g. email address) as the public key. It appears that, when used to for authenticating aggregation message, IBS allows the center to store 0 key since the verification keys are members' IDs. However, in a realistic group communication application, the center not only needs to verify that a message comes from an alleged sender, it also needs to ensure that the alleged sender is indeed a valid member of the group. In order to do so, the center needs to keep a list of valid member IDs online and compares each signature against it. This makes the effective center storage $O(n)$. In contrast, our scheme eliminates such overhead: group membership is self-authenticating through the aggregation data authentication tag. Besides IBS systems rely on a trusted key generation center (KGC) for extracting each user's private key, adding another layer of complexity.

Chapter 8

Conclusion

8.1 Summary of Contributions

The main contributions presented in this thesis are:

- We identified that a major performance bottleneck of existing private computation protocols is their heavy use of expensive public-key operations which impose a prohibitive constant factor in their complexity, although asymptotically they have the same complexity as their regular, non-private implementations. We show that a large number of useful algorithms can be implemented using addition-only steps which have simpler private implementations.

- We introduced the Peers for Privacy (P4P) framework for privacy-preserving distributed computation. P4P features a hybrid architecture between peer-to-peer (P2P) and the client-server model. This architecture exploits the heterogeneity of the players and greatly simplifies secure distributed computation.

- Within the P4P framework, the main computation is done using addition-only steps with secret sharing over *small* field. With this approach, private arithmetic operations on each player have the same cost as regular arithmetic and the algorithms can be implemented with the same real cost as their regular, non-private implementations.

- The P4P framework also features practical cheating users handling. We present very efficient probabilistic zero-knowledge protocols that can be used to verify the properties of user data such as equality and boundedness. Such protocols uses a linear number of inexpensive small field operations, and only a logarithmic number of large-field (1024 bits or more) cryptographic operations, achieving orders of magnitude reduction in running time over standard cryptographic techniques. These tools provide practical mechanisms to deal with cheating users and restrict their influences on the computation.

- We demonstrated the utility of the P4P framework using concrete examples from real-world applications. The examples include SVD, link analysis and association rule mining. We show that P4P implementation provides comparable or identical performance in terms of complexity and accuracy compared with their standard implementations.

- We also introduced scalable solutions for multicast encryption and bidirectional group communication. Our schemes feature strong security (IND-CCA2) and truly constant-size decryption keys whereas the lengths of both the encryption key and ciphertext are independent of group size. They provide secure data transmission support for the type of communication pattern required by the P4P framework and many other group-oriented applications.

8.2 Related Work

Theoretically, the data mining tasks performed by P4P can be done with secure multiparty computation (MPC) protocols, which provide general solutions for computing any probabilistic n-ary function among n players while protecting each player's private data [133, 61, 10, 60, 64]. The problem with the MPC protocols is their prohibitive cost. These protocols make heavy use of public-key cryptosystems or one-way functions, applying verifications or ZKPs at most steps, and some even operate at bit-level (rather than on arithmetic values). Even those arithmetic protocols typically have at least $O(sn^3)$ computation and communication complexity, with large hidden constant, where s is a security parameter. [1]

Existing privacy-preserving data mining solutions use either randomization (e.g. [50, 46]) or cryptographic techniques (e.g. [83, 45, 123, 132, 130]) to protect privacy. The first approach, besides sacrificing accuracy, has been shown to provide very little privacy protection in many cases [72]. Most of the cryptographic schemes use some form of MPC (e.g. [83, 45, 123] or homomorphic encryption [132, 130]. They attempted to obtain reasonable efficiency by targeting at specific problems or restricting the number of players.

P4P is also in the cryptographic category but it provides a more general primitive and much better efficiency. Its private vector addition protocol works in "normal" size (32 or 64 bits) fields so local computation is as efficient as regular arithmetics. All its zero-knowledge verifications involves only small (constant or $\log(m)$) number of big field operations thus adding very little cost.

A private SVD scheme was presented in [21], based on threshold homomorphic encryption and the conjugate gradient method of Polak-Ribiere [104]. Although reasonably efficient, their scheme works in big integer fields (1024 bits or more) and is not as practical as P4P.

8.3 Ongoing and Future Work

Our private summation protocol guarantees that no more information beyond the sums is revealed and we treat the sums as public. We have provided privacy analysis for the example algorithms in the early chapters of this thesis and showed that they do not cause privacy breach. There is ongoing work to provide a more general analysis and we have already obtained some initial results. [2] Basically, using the results in statistical database privacy [11, 43, 49], we are able prove that, under some conditions, the sums are in fact safe.

[1]For addition only functions some protocols only require $O(n)$ broadcasts (e.g. [34, 36]). However, broadcast has to be emulated with Byzantine agreement protocols in a realistic network such as the Internet. The most efficient broadcast protocols require $\Omega(n^2 l)$ bits to be sent for broadcasting an l-bit message [30].

[2]Since they are still preliminary we did not include them in this work.

Roughly speaking, these works showed that a noisy version of the sums could guarantee strong privacy provided that Tm is sub-linear in n, where T is the total number of iterations. In our settings, we are able to prove that, in some situations, external noise is *not* necessary for maintaining privacy. Instead, the randomness associated with an adversary's inherent uncertainty about unknown data is enough to prevent it from gaining significant information by seeing the aggregates. We can show that this protection is at least *equivalent* to that provided by the perturbation-based approach in [11, 43, 49].

Throughout this thesis, we have treated both the privacy peers and the server as passive adversaries. This is reasonable in many situations where the privacy peers enjoy a higher level of trust already endorsed in real-life (e.g. a union leader), or restricted by legislation or other factors (hospitals, research institutes, etc.), so that it is unlikely that they deviate from the protocol or try to bias the computation. This does not hold, however, when the privacy peers are from general user population. In this case we can use a random sampling method to verify the computation of privacy peers. Again the P4P architecture still uses the server to defend against external attacks but relies on redundancy to handle active cheating of privacy peers. We leave careful design and analysis of such mechanism as another future work.

We have implemented the basic components of the P4P framework, including the main computation, the L2-norm boundedness ZKP, and the private SVD scheme. The code is written in Java using a NativeBigInteger implementation from the I2P anonymous network (http://www.i2p.net/). The source code is released as a toolkit to the public and is available for download at http://www.cs.berkeley.edu/∼duan/research/p4p.html. We will continue to maintain the code and improve it as we obtain experience and feedback from users. In the near future, we plan to build more "middle tier" components to support more concrete applications. Our goal is to make P4P a useful tool for developers in areas such as data mining and others to build privacy preserving real-world applications.

Bibliography

[1] M. Abe, "Robust distributed multiplication without interaction," in *CRYPTO 1999*, ser. Lecture Notes in Computer Science, vol. 1666. Springer-Verlag, 1999, pp. 130–147.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of ACM SIGMOD Conference on Management of Data*, P. Buneman and S. Jajodia, Eds., Washington D.C., May 1993, pp. 207–216.

[3] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[4] ——, "Privacy-preserving data mining," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data.* New York, NY, USA: ACM Press, 2000, pp. 439–450.

[5] J. Anzai, N. Matsuzaki, and T. Matsumoto, "A quick group key distribution scheme with "entity revocation"," in *ASIACRYPT 1999*, ser. Lecture Notes in Computer Science, vol. 1716. Singapore: Springer, November 1999, pp. 333–347.

[6] D. Beaver and S. Goldwasser, "Multiparty computation with faulty majority," in *Proceedings of Advances in Cryptology – CRYPTO '89*, ser. Lecture Notes in Computer Science, vol. 435. Springer-Verlag, 1989, p. 589.

[7] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology.* London, UK: Springer-Verlag, 1996, pp. 1–15.

[8] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security.* ACM Press, 1993, pp. 62–73.

[9] ——, "Optimal asymmetric encryption – how to encrypt with RSA," in *EUROCRYPT 1994*, ser. Lecture Notes in Computer Science, vol. 950. Springer-Verlag, 1994, pp. 92–111.

[10] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC'88.* ACM, May 2–4 1988, pp. 1–10.

[11] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM Press, 2005, pp. 128–138.

[12] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Proceedings of Eurocrypt '2001*, ser. Lecture Notes in Computer Science, vol. 2045. Springer-Verlag, 2001, pp. 437–452.

[13] D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme," in *CRYPTO 1999*, ser. Lecture Notes in Computer Science, vol. 1666. Springer-Verlag, 1999, pp. 338–353.

[14] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *CRYPTO 2005*, ser. Lecture Notes in Computer Science, vol. 3621. Springer-Verlag, 2005, pp. 258–275.

[15] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology – EUROCRYPT 2000*, ser. Lecture Notes in Computer Science, vol. 1807. Springer-Verlag, 2000, pp. 431–444.

[16] C. Boyd, "Digital multisignatures," *Cryptography and Coding*, pp. 241–246, 1986.

[17] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Microsoft Research,, Tech. Rep. MSR-TR-98-12, October 1998.

[18] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *7th World-Wide Web Conference*, Brisbane, Australia, 1998.

[19] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *INFOCOMM'99*, 1999. [Online]. Available: citeseer.ist.psu.edu/article/canetti99multicast

[20] R. Canetti and S. Goldwasser, "An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack," in *EUROCRYPT 1999*, ser. Lecture Notes in Computer Science, vol. 1592. Springer-Verlag, 1999, pp. 90–106.

[21] J. Canny, "Collaborative filtering with privacy," in *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002, pp. 45–57. [Online]. Available: http://citeseer.nj.nec.com/canny02collaborative.html

[22] ——, "Collaborative filtering with privacy via factor analysis," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Tampere, Finland: ACM Press, 2002, pp. 238–245.

[23] J. F. Canny and Y. Duan, "Practical private computation of vector addition-based functions or: Can privacy be for free?" EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-12, February 8 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-12.html

[24] J. Carriere and R. Kazman, "Webquery: Searching and visualizing the web through connectivity," in *Proceedings of the International WWW Conference*, 1997.

[25] J. C. Cha and J. H. Cheon, "An identity-based signature from gap diffie-hellman groups," in *PKC '03: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography.* Springer-Verlag, 2003, pp. 18–30.

[26] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure internet multicast using boolean function minimization techniques," in *Proceedings IEEE Infocomm'99*, vol. 2, 1999, pp. 689–698. [Online]. Available: citeseer.ist.psu.edu/chang99key.html

[27] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *CRYPTO 1994*, ser. Lecture Notes in Computer Science, vol. 839. Springer-Verlag, 1994, pp. 257–270.

[28] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *Proceedings of IEEE Foundations of Computer Science*, 1985, pp. 383–395.

[29] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," in *NIPS 2006*, 2006.

[30] B. A. Coan and J. L. Welch, "Modular construction of a byzantine agreement protocol with optimal message bit complexity," *Information and Computation*, vol. 97, no. 1, pp. 61–85, 1992.

[31] W. W. Cohen, "Enron email dataset," http://www-2.cs.cmu.edu/˜enron/.

[32] A. Corrada-Emmanuel, "Enron email dataset research," http://ciir.cs.umass.edu/˜corrada/enron/.

[33] R. Cramer and I. Damgård, "Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free?" in *CRYPTO '98*, ser. Lecture Notes in Computer Science, vol. 1642. Springer-Verlag, 1998.

[34] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques.* Springer-Verlag, 2001, pp. 280–299.

[35] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *CRYPTO 1998*, ser. Lecture Notes in Computer Science, vol. 1462. Springer-Verlag, 1998, pp. 13–25.

[36] I. Damgård and J. B. Nielsen, "Universally composable efficient multiparty computation from threshold homomorphic encryption," in *CRYPTO 2003*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2003, pp. 247–264.

[37] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *WWW '07: Proceedings of the 16th international conference on World Wide Web.* New York, NY, USA: ACM Press, 2007, pp. 271–280.

[38] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to share a function securely," in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing.* ACM Press, 1994, pp. 522–533.

[39] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[40] H. Debar and A. Wespi, “Aggregation and correlation of intrusion-detection alerts,” in *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection.* Springer-Verlag, 2001, pp. 85–103.

[41] Y. Desmedt, “Society and group oriented cryptography: A new concept,” in *CRYPTO 1987*, ser. Lecture Notes in Computer Science, vol. 293. Springer-Verlag, 1987, pp. 120–127.

[42] Y. G. Desmedt and Y. Frankel, “Threshold cryptosystems,” in *CRYPTO 1989*, ser. Lecture Notes in Computer Science, vol. 435. Springer-Verlag, 1989, pp. 307–315.

[43] I. Dinur and K. Nissim, “Revealing information while preserving privacy,” in *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* New York, NY, USA: ACM Press, 2003, pp. 202–210.

[44] Y. Dodis and N. Fazio, “Public key trace and revoke scheme secure against adaptive chosen ciphertext attack,” in *Workshop on Public Key Cryptography – PKC '03*, ser. Lecture Notes in Computer Science, vol. 2567, 2003, pp. 100–115.

[45] W. Du, Y. Han, and S. Chen, “Privacy-preserving multivariate statistical analysis: Linear regression and classification,” in *SIAM International Conference on Data Mining*, 2004, pp. 222–233.

[46] W. Du and Z. Zhan, “Using randomized response techniques for privacy-preserving data mining,” in *KDD '03.* New York, NY, USA: ACM Press, 2003, pp. 505–510.

[47] Y. Duan and J. Canny, “How to construct multicast cryptosystems provably secure against adaptive chosen ciphertext attack,” in *RSA Conference 2006, Cryptographers' Track. San Jose, USA*, ser. Lecture Notes in Computer Science, vol. 3860. Springer-Verlag, 2006, pp. 244–261.

[48] Y. Duan, J. Wang, M. Kam, and J. Canny, “A secure online algorithm for link analysis on weighted graph,” in *Proceedings of the Workshop on Link Analysis, Counterterrorism and Security at the SIAM Data Mining Conference, 2005*, April 2005, pp. 71–81.

[49] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *TCC 2006*, ser. Lecture Notes in Computer Science, vol. 3876. Springer, 2006, pp. 265–284.

[50] A. Evfimievski, J. Gehrke, and R. Srikant, “Limiting privacy breaches in privacy preserving data mining,” in *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* New York, NY, USA: ACM Press, 2003, pp. 211–222.

[51] W. Fenner, “Internet group management protocol, version 2,” RFC-2236, November 1997.

[52] A. Fiat and M. Naor, "Broadcast encryption," in *CRYPTO 1993*, ser. Lecture Notes in Computer Science, vol. 773. Springer-Verlag, 1994, pp. 480–491.

[53] M. Fitzi, M. Hirt, and U. Maurer, "General adversaries in unconditional multi-party computation," in *Advances in Cryptology - ASIACRYPT 99*, ser. Lecture Notes in Computer Science, vol. 1716. Springer-Verlag, 1999, pp. 232–246.

[54] P.-A. Fouque and D. Pointcheval, "Threshold cryptosystems secure against chosen-ciphertext attacks," in *ASIACRYPT 2001*, ser. Lecture Notes in Computer Science, vol. 2248. Springer-Verlag, 2001, pp. 351–368.

[55] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP is secure under the rsa assumption," in *CRYPTO 2001*, ser. Lecture Notes in Computer Science, vol. 2139. Springer-Verlag, 2001, pp. 260–274.

[56] J. A. Garay, J. Staddon, and A. Wool, "Long-lived broadcast encryption," in *CRYPTO 2000*, ser. Lecture Notes in Computer Science, vol. 1880. Springer-Verlag, 2000, pp. 333–352.

[57] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust and efficient sharing of RSA functions," in *CRYPTO 1996*, ser. Lecture Notes in Computer Science, vol. 1109. Springer-Verlag, 1996, pp. 157–172.

[58] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified vss and fast-track multiparty computations with applications to threshold cryptography," in *PODC '98: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing.* ACM Press, 1998, pp. 101–111.

[59] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *CRYPTO'97*, ser. Lecture Notes in Computer Science. Springer-Verlag, 1997.

[60] O. Goldreich, *Foundations of Cryptography: Volume 2 Basic Applications.* Cambridge University Press, 2004.

[61] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game — a completeness theorem for protocols with honest majority," in *Proceedings of the 19th ACM Symposium on the Theory of Computing (STOC)*, 1987, pp. 218–229.

[62] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems." *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

[63] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[64] S. Goldwasser and L. Levin, "Fair computation of general functions in presence of immoral majority," in *Advances in Cryptology – CRYPTO '90*, ser. Lecture Notes in Computer Science, vol. 537. Springer-Verlag, 1991, pp. 77–93.

[65] D. Halevy and A. Shamir, "The LSD broadcast encryption scheme," in *CRYPTO 2002*, ser. Lecture Notes in Computer Science, vol. 2442. Springer-Verlag, 2002, pp. 47–60.

[66] H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) architecture," IETF Request for Comments, RFC 2094, July 1997.

[67] M. Hirt and U. Maurer, "Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract)," in *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 1997, pp. 25–34.

[68] ——, "Player simulation and general adversary structures in perfect multiparty computation," *Journal of Cryptology*, vol. 13, no. 1, pp. 31–60, 2000.

[69] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[70] S. Jarecki and A. Lysyanskaya, "Adaptively secure threshold cryptography: Introducing concurrency, removing erasures (extended abstract)," in *Proceedings of Eurocrypt 2000*, ser. Lecture Notes in Computer Science, vol. 1807. Springer-Verlag, 2000, pp. 221–242.

[71] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1026–1037, 2004.

[72] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques," in *ICDM '03*. Washington, DC, USA: IEEE Computer Society, 2003, p. 99.

[73] H. Kautz, B. Selman, and A. Milewski, "Agent amplified communication," in *AAAI-96*. Cambridge, Mass.: MIT Press, 1996, pp. 3–9, portland, Oreg.

[74] H. Kautz, B. Selman, and M. Shah, "Combining social networks and collaborative filtering," *Comm. ACM*, vol. 40, no. 3, pp. 63–65, 1997.

[75] M. Kearns, "Efficient noise-tolerant learning from statistical queries," in *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 1993, pp. 392–401.

[76] C. H. Kim, Y. H. Hwang, and P. J. Lee, "An efficient public key trace and revoke scheme secure against adaptive chosen ciphertext attack," in *ASIACRYPT 2003*, ser. Lecture Notes in Computer Science, vol. 2894. Springer-Verlag, 2003, pp. 359–373.

[77] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999. [Online]. Available: citeseer.nj.nec.com/kleinberg99authoritative.html

[78] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of the acquisition induction and representation of knowledge," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.

[79] R. B. Lehoucq and J. A. Scott, "An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices," Preprint MCS-P547-1195, Argonne National Laboratory, Argonne, Ill, 1996.

[80] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, 1998.

[81] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch rekeying for secure group communications," in *Proceedings of the tenth international World Wide Web conference on World Wide Web*, Orlando, FL USA, 2001, pp. 525–534. [Online]. Available: citeseer.ist.psu.edu/li01batch.html

[82] C. H. Lim and P. J. Lee, "Another method for attaining security against adaptively chosen ciphertext attacks," in *CRYPTO 1993*, ser. Lecture Notes in Computer Science, vol. 773. Springer-Verlag, 1993, pp. 420–434.

[83] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Journal of cryptology*, vol. 15, no. 3, pp. 177–206, 2002.

[84] D. Liu, P. Ning, and K. Sun, "Efficient self-healing group key distribution with revocation capability," in *Proceedings of the 10th ACM conference on Computer and communication security.* ACM Press, 2003, pp. 231–240.

[85] M. Luby and J. Staddon, "Combinatorial bounds for broadcast encryption," in *EUROCRYPT 1998*, ser. Lecture Notes in Computer Science, vol. 1403. Springer-Verlag, 1998, pp. 512–526.

[86] D. W. MacDonald and M. S. Ackerman, "Just talk to me: A field study of expertise location," in *ACM CSCW-98*, 1998, pp. 315–324.

[87] J. Markus and J. Ari, "Millimix: Mixing in small batches," Tech. Rep., 1999.

[88] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, ser. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1996.

[89] R. Motwani and P. Raghavan., *Randomized Algorithms.* Cambridge University Press, 1995.

[90] M. J. Moyer, J. R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network Magazine*, vol. 13, no. 6, pp. 12–23, November/December 1999.

[91] D. Naor, M. Naor, and J. B. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *CRYPTO 2001*, ser. Lecture Notes in Computer Science, vol. 2139. Springer-Verlag, 2001, pp. 41–62.

[92] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proceedings of the twenty-second annual ACM symposium on Theory of computing.* ACM Press, 1990, pp. 427–437.

[93] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," in *Proceedings of Financial Crypto 2000*, 2000.

[94] A. Newell and P. S. Rosenbloom, "Mechanisms of skill acquisition and the law of practice," in *J.R. Anderson (Ed.), Cognitive Skills and their Acquisition (pp. 1-55). Hillsdale, NJ: Earlbaum*, 1981.

[95] A. Newell and P. Rosenbloom, *Matrix Computations.* Johns Hopkins University Press, 1989.

[96] A. Y. Ng, A. X. Zheng, and M. Jordan, "Stable algorithms for link analysis," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM Press, 2001, pp. 258–266.

[97] A. Y. Ng, A. X. Zheng, and M. I. Jordan, "Link analysis, eigenvectors and stability," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, August 2001, pp. 903–910.

[98] P. Paillier, "Public-key cryptosystems based on discrete logarithms residues," in *EUROCRYPT 1999*, ser. Lecture Notes in Computer Science, vol. 1592. Springer-Verlag, 1999, pp. 223–238.

[99] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology – CRYPTO '91*, ser. Lecture Notes in Computer Science, vol. 576. Springer-Verlag, 1991, pp. 129–140.

[100] A. Perrig, R. Canetti, D. Tygar, and D. Song, "Efficient authentication and signature of multicast streams over lossy channels," in *IEEE Symposium on Security and Privacy*, May 2000, pp. 56–73.

[101] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and secure source authentication for multicast," in *NDSS01*, 2001.

[102] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, 2002.

[103] P. Pirolli, J. Pitkow, and R. Rao, "Silk from a sow's ear: Extracting usable structures from the web," in *Proc. ACM Conf. Human Factors in Computing Systems, CHI.* ACM Press, 1996. [Online]. Available: citeseer.nj.nec.com/pirolli96silk.html

[104] E. Polak, *Computational Methods in Optimization.* Academic Press, 1971.

[105] C. Rackoff and D. R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *CRYPTO 1991*, ser. Lecture Notes in Computer Science, vol. 576. Springer-Verlag, 1992, pp. 433–444.

[106] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *Proceedings of the 6th ACM conference on Computer and communications security.* ACM Press, 1999, pp. 93–100.

[107] RSA Labs, "PKCS#1 v2.1: RSA cryptography standard," 2002.

[108] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[109] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system – a case study," in *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000, full length paper.

[110] M. F. Schwartz and D. C. M. Wood, "Discovering shared interests using graph analysis," *Comm. ACM*, vol. 36, no. 8, pp. 78–89, 1993.

[111] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A scalable group re-keying approach for secure multicast," in *IEEE Symposium on Security and Privacy*, 2000, pp. 215–228. [Online]. Available: citeseer.ist.psu.edu/setia00krono.html

[112] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612–613, November 1979.

[113] ——, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology.* Springer-Verlag New York, Inc., 1985, pp. 47–53.

[114] M. Shehab, E. Bertino, and A. Ghafoor, "Efficient hierarchical key generation and key diffusion for distributed sensor networks," in *IEEE SECON 2005*, 2005.

[115] V. Shoup, "OAEP reconsidered," in *CRYPTO 2001*, ser. Lecture Notes in Computer Science, vol. 2139. Springer-Verlag, 2001, pp. 239–259.

[116] V. Shoup and R. Gennaro, "Securing threshold cryptosystems against chosen ciphertext attack," *J. Cryptology*, vol. 15, no. 2, pp. 75–96, 2002.

[117] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean, "Self-healing key distribution with revocation," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy.* IEEE Computer Society, 2002, p. 241.

[118] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory.* Academic Press, 1990.

[119] G. Strang, *Linear Algebra and Its Applications, 2nd Edition.* Academic Press, 1980.

[120] Y. Sun and K. R. Liu, "Securing dynamic membership information in multicast communications," in *Proceedings IEEE INFOCOMM '04*, 2004.

[121] W.-G. Tzeng and Z.-J. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography.* Springer-Verlag, 2001, pp. 207–224.

[122] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* New York, NY, USA: ACM Press, 2002, pp. 639–644.

[123] ——, "Privacy-preserving k-means clustering over vertically partitioned data," in *KDD '03.* New York, NY, USA: ACM Press, 2003, pp. 206–215.

[124] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 9, pp. 1614–1631, September 1999.

[125] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," IETF Request For Comments, RFC 2627, June 1999.

[126] H. Wang, "Resilient lkh: Secure multicast key distribution schemes," in *Proceedings of the 2003 International Workshop on Advanced Developments in Software and Systems Security (WADIS)*, 2003.

[127] C. K. Wong and S. S. Lam, “Digital signatures for flows and multicasts,” Tech. Rep., 1998.

[128] C. K. Wong, M. Gouda, and S. S. Lam, “Secure group communications using key graphs,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, 2000.

[129] C. K. Wong and S. S. Lam, “Keystone: A group key management service,” in *International Conference on Telecommunications, ICT 2000*, 2000. [Online]. Available: citeseer.ist.psu.edu/wong00keystone.html

[130] R. Wright and Z. Yang, “Privacy-preserving bayesian network structure computation on distributed heterogeneous data,” in *KDD ’04*. New York, NY, USA: ACM Press, 2004, pp. 713–718.

[131] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, “Reliable group rekeying: a performance analysis,” in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM Press, 2001, pp. 27–38.

[132] Z. Yang, S. Zhong, and R. N. Wright, “Privacy-preserving classification of customer data without loss of accuracy,” in *SDM 2005*, 2005.

[133] A. C.-C. Yao, “Protocols for secure computations,” in *FOCS ’82.* IEEE, 1982, pp. 160–164.

[134] J. Zhan, “Privacy preserving collaborative data mining,” Ph.D. dissertation, University of Ottawa, 2006.

[135] J. Zhan, S. Matwin, and L. Chang, “Privacy-preserving collaborative association rule mining,” in *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security.*