Social-Aware Decentralization for Secure and Scalable Multi-Party Computations

Yuzhe Tang
Dept. of EECS, Syracuse University
ytang100@syr.edu

Sucheta Soundarajan
Dept. of EECS, Syracuse University
susounda@syr.edu

***Abstract*—This work studies the problem of *MPC decentralization* - that is, identifying a set of computing nodes to securely and efficiently execute the multi-party computation protocol (MPC) over a sensitive dataset. To balance between under-decentralization with high risk and over-decentralization with high cost, our unique approach is to add social-awareness, that is, the MPC protocol, running over a social network, is properly decentralized among the computing nodes selected carefully based on their social relationship. The key technical challenge is in estimating the risk of collusion between nodes on whom the computation is run. We propose solutions to estimate the risk of collusion based on (incomplete) social relationship, as well as algorithms for finding the MPC nodes such that the risk of collusion is minimized. We evaluate our methods on several real-world network datasets, and show that they are effective in minimizing the risk levels. This work has potential in enabling efficient privacy-preserving data sharing and computation in emerging big-data federation platforms, in healthcare, financial marketplaces, and other application domains.**

I. Introduction

In the age of big data, owners of sensitive data may autonomously join and form an emerging data-federation platform, referred to as a private-information network (PIN). For instance, Health-care Information Networks emerge at the levels of federal government [1], states [2], and regions [3], where different hospitals and clinical centers connect local databases of patient electronic health records in the hope of sharing data for better patient treatment. PINs emerge for many other new applications, such as global enterprise computing (e.g. Google's Spanner [4] databases across multiple countries [5]), cross-department data sharing in SmartCity, genome data-sharing (e.g. Beacon Network[1]), and financial data exchange [6].

The value of these private information networks stems from the global dataset and the possible new knowledge or insights that can be extracted from them. For instance, the global dataset of patient medical records is stored across many hospitals, and allows one to obtain the *full* medical history of the patient of interest.

The key barrier to realizing the value of PINs is the privacy concern on sharing data across the trust boundary. In a PIN, data sharing has to be in compliance with data protection laws (HiPAA[2]) and ethically speaking, it should also respect data owners' concerns on possible privacy leakage. The protocols of secure multi-party computation (MPC [7], [8]) provide a promising approach to protect data privacy in PINs. Briefly, the purpose of an MPC is to evaluate some function on multiple private inputs held by different parties, and to disclose only the final output without leaking the private inputs (and any intermediate results).

In this paper, we address the problem of decentralizing MPC across nodes in a social network so as to minimize the risk of collusion between nodes while retaining performance efficiency. There is a tradeoff between risk minimization and performance efficiency. On the one hand, when the MPC is executed on a small group of nodes (i.e. under-decentralized), the risk of convincing all computing nodes to collude is high but running the MPC at small scale leads to efficient performance. On the other hand, overly decentralizing the MPC to the entire large-scale social network can sufficiently decentralize trust and minimize the risk of collusion, but at the expense of increasing performance overhead. To strike a balance between the two ends, our observation is that the tradeoff is not linearly proportional; for instance, scaling down an overly-decentralized MPC does not *necessarily* lead to increased risk of collusion. If one knows social relationship, the unnecessary decentralization can be avoided on social friends who might collude. Hence for the best efficiency-security tradeoff, the decentralization is applied only to non-friends whose chance to collude is low. In other words, the awareness of social relationship presents us a "free" performance optimization opportunity, in the sense of being free of security degradation.

The key challenge in realizing the social-aware MPC decentralization is uncertainty nature of social user behavior, particularly in terms of forming collusion. We propose to model the risk of collusion by the likelihood of two probabilistic events both happening: 1) each MPC node being dishonest, and 2) MPC nodes being acquaintance, thus making it possible for them to "collaborate." The latter factor is further modeled by the "closeness" of the MPC nodes in the social network – the intuition is that if MPC nodes are acquaintances or closely connected, it is more likely that they will collude. We propose a fully quantifiable framework for modeling the risk of collusion and applying the model to the problem of MPC decentralization.

While there is a large body of research on secure multi-party computations [7], [**?**], [8], [**?**], [**?**], [9], [10], there are

[1]https://genomicsandhealth.org/files/public/Beacon-FAQ.pdf
[2]http://www.gpo.gov/fdsys/pkg/CFR-2010-title45-vol1/pdf/CFR-2010-title45-vol1-sec164-502.pdf

relatively few works focusing on minimizing the *use* of MPC with awareness to application features, such as social trust relationships. To the best of our knowledge, this is the first work that leverages the social connections to minimize the risk of collusion and the use of MPC for privacy-preserving big-data computation.

II. Research Formulation

A. Motivating application scenarios

Our proposed work on MPC decentralization can be motivated by the applications of running secure computations in real-world large-scale P2P networks.

In *P2P and grid computing*, a data owner of a sensitive dataset wants to leverage the "free" resources in P2P computation networks (e.g. SETI@Home[3], Open Science Grid[4]), but is reluctant to trust any specific *individual* peer who can be arbitrary entity joining the ad-hoc P2P network. Instead, it may be more practical to decentralize the trust among a selected group of peers and execute the secure computation of her data in the group.

In *distributed social networks*, such as Diaspora[5] and Twister[6], social users are registered at their "home" host servers which run autonomously and in different locations. The host servers may want to collectively conduct joint analysis over the global social dataset, yet they do not necessarily trust each other in sharing their social user data. Running the joint analysis on a single trusted third party (TTP) results in centralized strong trust which is difficult in real life. Running the joint analysis on *all* host servers, while effectively decentralizing the trust, results in high performance overhead. Appropriately decentralizing the trust, which does not sacrifice security under collusion attacks, would greatly improve the performance. Such proper level of trust decentralization can be made possible by the awareness of the social relationships; for instance, decentralizing the computation between two social nodes is only necessary when the two nodes/users do not have social relationship.

In Cryptocurrency (e.g. BitCoin[7]), the *public Blockchain P2P network* runs in a marketplace among buyers and sellers and its miner subnetwork functions as transaction ledger and arbitrator (by running a consensus protocol). The buyer-seller relationship can be leveraged to execute the multi-party Blockchain consensus computation among a subset of miners, rendering efficiency. The intuition is that running the protocol in a selected subnetwork may achieve the same security level against collusion attacks with running that in the entire network, while the cost of the former is much lower.

In *the social web browsers* (e.g. Chrome where the browser is bundled with social user identity), end users' data are collected and statistics about them are calculated in a privacy-preserving fashion [11]. Decentralizing the statistics computation among the entire "Chrome" user population would be prohibitively expensive and unnecessary even in security. Given the social relationship, decentralizing the secure computation can be constraint to a selected sub-population such that the security under collusion attacks is not sacrificed while it improves performance and scalability.

While there are many other similar scenarios, the common paradigm is that secure computation (in confidentiality and/or integrity) over sensitive data is executed on a decentralized P2P computing platform where the "social" relationship between nodes/users is known. Centralized trust by running the computation on a single node is impractical and unsecured. Overly decentralized trust by running the computation over the entire large-scale network is also impractical in the sense of performance inefficiency and inscalability. By awareness of the trust relationship, our goal is to find a subset of social nodes to execute the secure P2P computation (or MPC) in such a way that the scalability is improved while retaining the same level of security with the entire-network computation. In the following, we formally describe the model of our system to pave the way of formulating the MPC-decentralization problem.

B. The MPC-Decentralization Problem

Data/system model: In our system model, there is a network of N social sites G_N, each representing one or more social users. There are social edges between the sites representing the underlying social network.

In our data model, the social graph among the N social sites is publicly known, though the social users' profile data is not necessarily public. Meanwhile, there is private data on which the secure computation is run. The private data can be any arbitrary dataset depending on the application. For instance, in the privacy-preserving distributed social networks, the private data is social users' data itself. In the P2P networks, it is the owner's private data.

MPC decentralization is about selecting a subgraph of the social network to run the MPC protocol on the private dataset. Suppose an MPC subgraph S of M nodes is selected from G_N. When running MPC on S, the private data is split into M shares and sent to the M nodes. Then a protocol of MPC, such as GMW [8], will be executed on the M shares/sites.

While there are exponentially many ways to choose S given G_N, we aim at producing a "good" MPC subgraph in two senses: First , it should not be risky to execute an MPC; in other words, it is unlikely that all nodes in the MPC subgraph will collude, so the security of MPC can be ensured. Second, the cost of running MPC in the selected subgraph should be affordable. The cost of running MPC is an active research topic of its own [12], and we use the simplest heuristic in this paper, which is to bound the number of nodes to be smaller than a predefined threshold value, C. We focus on modeling the risk component in this work. To do that, we formalize our security definition.

[3]https://setiathome.berkeley.edu/
[4]https://www.opensciencegrid.org/
[5]https://www.joindiaspora.com/
[6]http://twister.net.co/
[7]https://bitcoin.org/en/

Security definition: In the MPC phase, we consider the semi-honest model for nodes in S, where each party follows the prescribed computation process but is curious about the data flowing through the process. Such intermediate-result data is used to guess what (sensitive) value the original input takes. MPC nodes in S (or more generally G_N) may collude to gain advantage in improving the probability of correctly guessing the data values. Our security goal is that any social user (either participating in MPC or not) who is semi-honest and observes computation data, cannot gain non-negligible advantage in guessing the secret input values. This follows the classic computational security of MPC [13]. In other words, our security goal is to meet the assumptions made in MPC protocol.

One key assumption in an MPC protocol is regarding how much collusion the protocol can tolerate. While different protocols are designed to tolerate different levels of colluding attacks, all protocols are broken if all nodes in S collude. This condition is the basis of modeling risk in our problem. That is, minimizing the risk of decentralizing MPC in S is modeled by minimizing the probability that all nodes in S collude.

Ideally, when it is known which subset of nodes in G_N are colluding, calculating the collusion probability w.r.t. S becomes trivial, which is as simple as determining if all nodes in S are among these colluding nodes. However, in reality, the set of collusion nodes are unknown and hidden from the public. With the only known information being the social graph structure, the collusion probability can be estimated by following intuition: Each social user has her own probability of being dishonest and collusion occurs only among dishonest social users. *The more closely connected these dishonest social users are in the social graph, the more likely a collusion is formed among them.*

III. Proposed Approach

In this section, we propose a collusion model that uses node honesty probabilities as well as the structure of the chosen subgraph to estimate the risk of collusion. We assume that each node u has an associated weight $w_u = -\log Pr[u \text{ is dishonest}]$ representing the probability that u is dishonest. If nodes act independently, the probability of all nodes in a subgraph being dishonest is thus given by $\Pi_u Pr[u \text{ is dishonest}] = 2^{-\sum_u w_u}$.

However, because it is not reasonable to believe that nodes make collusion decisions independently, we define the *collusion score* of subgraph S_N as its density. The density of a subgraph is the number of edges in the subgraph divided by the maximum possible number of edges in the subgraph ($\frac{1}{2}n(n-1)$, where n is the number of nodes in the subgraph). Intuitively, nodes in a high-density subgraph are well-connected to one another, and so are more likely to collude. We thus aim to find a low-density subgraph to minimize the risk of collusion. Formally, we model the MPC decentralization problem as the Minimum-Density Network-MPC problem, denoted by MDN-MPC.

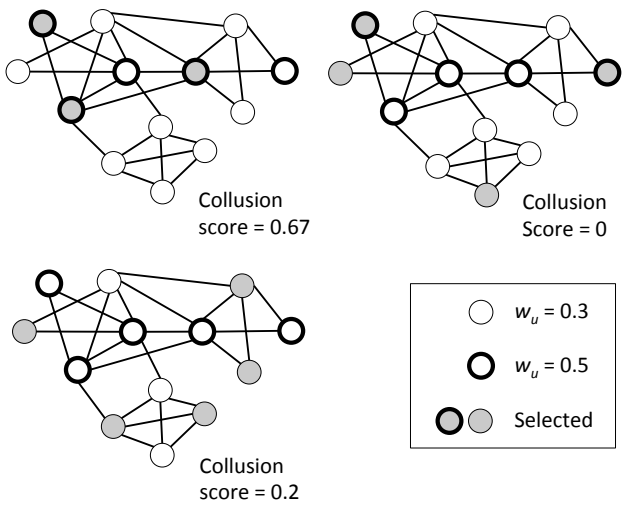

Fig. 1. An example with R = 1.5, C = 4. By choosing nodes appropriately, one can select a subgraph with collusion score 0.

Minimum-Density Network MPC (MDN-MPC): Suppose that one is given a graph G_N where each node u has weight w_u, and constants $C, R > 0$. Find a set $S_M = \{u_1, ..., u_M\}$, where $n \leq C$ and $w_{u_1} + ... + w_{u_M} \geq R$, such that the collusion score of the subgraph induced by S_M in G_N is minimized.

Here, C represents the cost constraint: we cannot divide the computation among too many nodes, or communication costs become prohibitively large. R represents a budget of risk of nodes being dishonest; in total, the joint probability that all nodes are dishonest should be bounded by R. That is, $\sum_u w_u \geq R$ is equivalent to $\Pi_u Pr[u \text{ is dishonest}] \leq 1/2^R$.

First, we show that the MDN-MPC problem is NP-Hard. Thus, it is not possible to create a fast algorithm to solve this problem exactly. We then present three heuristics: The first is a naive method that does not take graph structure into account. The second method is a greedy algorithm that updates its evaluation of each node in each iteration. As we will see in Section IV, this method is effective but slow. The third method is a faster greedy algorithm that performs fewer updates, and is faster but still effective.

A. *Hardness of* MDN-MPC

Theorem: MDN-MPC is NP-Hard.

Proof: We prove that MDN-MPC is NP-Hard via a reduction from Independent Set. The Independent Set problem is as follows: Given an undirected graph G and an integer $k > 0$, is there a set of k nodes $H \in G$ such that no two nodes in H are adjacent to one another? Independent Set is known to be NP-Complete.

Independent Set reduces to MDN-MPC in polynomial time as follows: Suppose one is given a graph G_{IS} and constant k as input to Independent Set. Suppose that one has an algorithm to solve MDN-MPC, which takes as input a weighted graph G_{MDN}, and constants $C, R > 0$. Set $G_{MDN} = G_{IS}$, and set all node weights equal to 1. Set

$R, C = k$. Then G_{IS} has an independent set of size k if and only if the set found by the MDN-MPC algorithm has density 0.

Because all nodes in G_{MDN} have weight 1, in order for the MDN-MPC algorithm to meet the risk constraint given by $R = k$, it must select at least k nodes. The constraint $C = k$ ensure that it will select no more than k nodes. The algorithm will thus find the minimum-density set of k nodes. Density cannot be less than 0, and a set of density 0 has no edges, so is an independent set. Thus, if the graph contains an independent set of size k, the graph contains a set of size k with density 0. The MDN-MPC algorithm will thus find a 0-density set of k nodes, which is an independent set of size k. Likewise, if the MDN-MPC algorithm finds a set of density 0, this set is an independent set. □

B. Algorithms for MDN-MPC

First, suppose that we remove the density requirement from the MDN-MPC problem, so we just wish to find a set of at most C nodes with at least R total weight. This can trivially be solved by Algorithm 1:

Algorithm 1: Naive Greedy. *Sort the nodes in descending order by their weight. Return the first C nodes in this list.*

The **Naive Greedy** algorithm serves as a baseline, in the sense that it provides an upper bound on the density we should expect.

Algorithm 2: Slow Greedy. *Initialize set $S = \emptyset$. In each iteration i, calculate R_i, the weight still needed to meet the risk constraint (given by R minus the sum of the w_u values so far), and C_i, the amount of nodes that may still be added (given by C minus the number of nodes currently in S). For each node u not in S, if $w_u \geq \frac{R_i}{C_i}$, calculate d_u, the number of connections u has to nodes in S. Let $r_u = \frac{w_u}{R_i d_u}$. Find the node with the largest r_u value, and add it to S.*

The **Slow Greedy** algorithm builds set S one node at a time. In each iteration, it calculates the required minimum average weight of a node added to the set as follows: If R_i weight is still needed to meet the R risk constraint, and only C_i nodes can still be added to S, then the added nodes must have at least $\frac{R_i}{C_i}$ weight on average. In each iteration, the algorithm generates a list of candidate nodes u with weight $w_u \geq \frac{R_i}{C_i}$. It then selects the candidate node with the highest r_u score, which is calculated from its w_u value as well as the number of connections that it has into the current set S (fewer connections are better). This algorithm finds low-density sets, but because the node scores must be updated after every iteration, it can be slow.

Algorithm 3: Fast Greedy. *For each node u, calculate u's degree d_u (i.e., the number of neighbors that u has), and how much it can contribute toward the risk constraint, $\frac{w_u}{R}$. Calculate the ratio r_u of its contribution to its degree: $\frac{w_u}{Rd_u}$. Sort the nodes in descending order of their r_u scores. Let u_j represent the node in position j of this list. Iterate through this list. In each iteration i, calculate R_i, the amount of weight still needed, and C_i, the number of nodes that can still be added (given by C minus the number of nodes added so far). If $w_{u_j} \geq R_i/C_i$, add u_j to the set of selected nodes.*

The **Fast Greedy** algorithm is similar to the **Slow Greedy** algorithm, except that instead of calculate the number of connections that a node has into the current set S, it only looks at each node's total degree d_u. This way, each node is scored only once, allowing for a substantial improvement in running time.

IV. Experiments

In this section, we describe our experimental results. We see that the **Naive Greedy** method runs very quickly, but because it does not take network structure into account, fails to find sets of nodes with low collusion scores. The **Slow Greedy** method is very effective at finding sets of nodes with low collusion scores, but is extremely slow on large networks. The **Fast Greedy** method finds sets of nodes that are almost as good as those found by **Slow Greedy**, but is much faster.

A. Experimental Setup

Datasets: We consider the following four datasets: **Grad** and **Ugrad** are portions of the Facebook network corresponding to students at Rice University.[8] **Grad** has 503 nodes and 6,512 edges, and **Ugrad** has 1,220 nodes and 86,416 edges. **Email** is an e-mail network corresponding to a European research institution. It has 986 nodes and 16,064 edges. **Astro** is a scientific collaboration network, with edges representing paper co-authorship. It has 18,772 nodes and 198,110 edges.[9]

Experimental Settings: We simulate dishonesty probabilities in two ways: uniformly at random over the interval [0.01, 0.99] and by the normal distribution with a mean of 0.5 and standard deviation of 0.2, trimmed to the interval [0.01, 0.99]. We then assign node weights by taking the negative log of the dishonesty probabilities. A high weight indicates that a node has a low probability of being dishonest, and so is good to include.

We set C (the maximum number of nodes selected) to be one half of the number of nodes in the graph and R (the risk constraint) to be one quarter of the number of nodes in the graph (though one may end up selecting more or fewer nodes than this, depending on their weights).

B. Results

For each algorithm, dataset, and risk distribution (uniform random or normal), we present (1) the total number of nodes selected, (2) the density of the resulting subgraph, and (3) the running time of the algorithm. Results are summarized in Tables I and II.

From these results, we observe that the **Naive Greedy** method, as expected, is very fast, but because it does not use graph structure, it selects nodesets with high collusion scores; in other words, the nodes selected by **Naive Greedy** meet the risk and cost constraints, but are likely to collude. The **Slow Greedy** method finds extremely sets of nodes with

[8] Obtained from Alan Mislove.

[9] **Email, Astro** obtained from http://snap.stanford.edu.

Network	C	R	Method	Num. Nodes	Collusion Score	Run. Time (s)
Grad	251	125	**Naive Greedy**	59	3.0e-2	0.02
			Slow Greedy	77	0.0	0.10
			Fast Greedy	98	5.2e-3	0.02
Ugrad	610	305	**Naive Greedy**	161	5.7e-2	0.39
			Slow Greedy	248	6.9e-3	1.4
			Fast Greedy	226	1.2e-2	0.38
Email	493	246	**Naive Greedy**	144	3.2e-2	1.2
			Slow Greedy	444	1.7e-3	2.1
			Fast Greedy	191	0.0	1.7
Astro	9,386	4.693	**Naive Greedy**	2,513	1.0e-3	2.1
			Slow Greedy	2,985	1.1e-6	202.4
			Fast Greedy	4,146	1.0e-4	2.37

TABLE I
EXPERIMENTAL RESULTS FOR NORMALLY-DISTRIBUTED DISHONESTY PROBABILITIES

Network	C	R	Method	Num. Nodes	Collusion Score	Run. Time (s)
Grad	251	125	**Naive Greedy**	40	2.4e-2	0.02
			Slow Greedy	43	0.0	0.06
			Fast Greedy	62	5.3e-3	0.03
Ugrad	610	305	**Naive Greedy**	98	4.9e-2	0.35
			Slow Greedy	171	2.4e-3	0.98
			Fast Greedy	146	1.1e-2	0.47
Email	493	246	**Naive Greedy**	78	4.1e-2	1.3
			Slow Greedy	94	0.0	1.4
			Fast Greedy	171	4.8e-4	1.4
Astro	9,386	4.693	**Naive Greedy**	1,483	1.0e-3	1.9
			Slow Greedy	1,625	2.3e-6	96.7
			Fast Greedy	2,483	9.4e-5	2.0

TABLE II
EXPERIMENTAL RESULTS FOR UNIFORMLY RANDOM-DISTRIBUTED DISHONESTY PROBABILITIES.

extremely low collusion scores that are usually at least an order of magnitude lower than those found by the baseline **Naive Greedy** method. But **Slow Greedy** is prohibitively slow: for the **Astro** dataset, containing 18,772 nodes, **Slow Greedy** took up to over 3 minutes to select appropriate nodes. This method will clearly be inappropriate for networks with hundreds of thousands or millions of nodes. The **Fast Greedy** method is usually almost as fast, or faster than, the **Naive Greedy** method, but still finds sets of nodes with low collusion scores. These results hold across both types of risk distribution.

V. Related Work

Secure multi-party computation (MPC) has been a focus of theoretic research since the 80s [7], [8], and the last decade has seen a rise in systems-oriented research on MPC. This body of research supports programming tools and runtime systems for MPC, such as MPC compilers [14], [15] and MPC execution engines [16], [17]. Such systems usually follow proven secure protocols for MPC, such as Yao's Garbled Circuit [7], GMW [8], SPDZ [9], etc. Privacy-preserving data mining [18], [19] leverages domain-specific construct to secure a certain kind of data-mining computation. In particular, privacy-preserving indexes and locator services are proposed to redirect queries to the end data owners in a secure way [20], [21]. Our research builds on top of MPC systems with a particular concern on exploiting query-specific semantics.

Another line of related research is designing scalable P2P networks. A graph-theoretic approach is to design a scalable overlay on the distributed hash table abstraction (DHT) of the P2P network. Many proposed DHT overlays, such as Chord [22], are based on Plaxton Mesh topology. Others, such as CAN [23] leverages d-torus topology. Data management layer, such as secondary-index support [24], has been added to the key-value store of DHT, in the hope of supporting more complex and expressive computations such as range queries [25], multi-dimensional queries [26], [27], etc.

VI. Conclusions

We have considered the problem of decentralizing MPC across nodes in a social network, with the goal of minimizing the risk of collusion between nodes. To our knowledge, we are the first to consider this problem, which has applications in areas such as peer-to-peer computing and privacy-preserving distributed social networks. We presented a model in which each node is assigned a weight related to its probability of being dishonest, and then presented the MINIMUM-DENSITY NETWORK MPC (MDN-MPC) problem.

In this problem, one is given a positively-weighted graph G, and positive constants C and R. The goal is to select no more than C nodes from G such that the sum of their weights is at least R, and the density of the subgraph induced by the nodes is minimized. We presented a fast greedy algorithm to find appropriate subgraphs, and showed that it achieved good results on several real datasets.

This work is preliminary, and we hope that it encourages other work along these lines. There are many open topics to explore. For instance, there are many alternatives to density that one might use, including path-based characteristics, clustering of the nodes, etc. Many of the problem formulations suggested by these alternatives are likely to be NP-Hard, but there may be provably-good greedy algorithms for approximating the optimal solution. We are additionally interested in exploring problems in which there are additional constraints on the nodes, such as minimum total resource requirements.

References

[1] "Nwhin: http://www.hhs.gov/healthit/healthnetwork."
[2] "Shin-ny: http://www.health.ny.gov/technology/projects/."
[3] "Healtheconnections rhio: http://www.healtheconnections.org/rhio."
[4] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. C. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, vol. 31, no. 3, p. 8, 2013. [Online]. Available: http://doi.acm.org/10.1145/2491245
[5] A. Vulimiri, C. Curino, P. B. Godfrey, T. Jungblut, J. Padhye, and G. Varghese, "Global analytics in the face of bandwidth and regulatory constraints," in *12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015*, 2015, pp. 323–336. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/vulimiri

[6] A. Narayan, A. Papadimitriou, and A. Haeberlen, "Compute globally, act locally: Protecting federated systems from systemic threats," in *10th Workshop on Hot Topics in System Dependability, HotDep '14, Broomfield, CO, USA, October 5, 2014.*, 2014. [Online]. Available: https://www.usenix.org/conference/hotdep14/workshop-program/presentation/narayan

[7] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, 1986, pp. 162–167. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1986.25

[8] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, A. V. Aho, Ed. ACM, 1987, pp. 218–229. [Online]. Available: http://doi.acm.org/10.1145/28395.28420

[9] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits," in *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, 2013, pp. 1–18. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40203-6_1

[10] G. Cohen, I. B. Damgård, Y. Ishai, J. Kölker, P. B. Miltersen, R. Raz, and R. D. Rothblum, "Efficient multiparty protocols via log-depth threshold formulae - (extended abstract)," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, 2013, pp. 185–202. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40084-1_11

[11] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 1054–1067. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660348

[12] A. Ben-Efraim, Y. Lindell, and E. Omri, "Optimizing semi-honest secure multiparty computation for the internet," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 578–590. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978347

[13] O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[14] A. Rastogi, M. A. Hammer, and M. Hicks, "Wysteria: A programming language for generic, mixed-mode multiparty computations," in *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 2014, pp. 655–670. [Online]. Available: http://dx.doi.org/10.1109/SP.2014.48

[15] E. M. Songhori, S. U. Hussain, A. Sadeghi, T. Schneider, and F. Koushanfar, "Tinygarble: Highly compressed and scalable sequential garbled circuits," in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, 2015, pp. 411–428. [Online]. Available: http://dx.doi.org/10.1109/SP.2015.32

[16] X. S. Wang, Y. Huang, T. H. Chan, A. Shelat, and E. Shi, "SCORAM: oblivious RAM for secure computation," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 191–202. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660365

[17] S. G. Choi, K. Hwang, J. Katz, T. Malkin, and D. Rubenstein, "Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces," in *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012*, 2012, pp. 416–432.

[18] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, 2002, pp. 639–644. [Online]. Available: http://doi.acm.org/10.1145/775047.775142

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

[20] Y. Tang and L. Liu, "Privacy-preserving multi-keyword search in information networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2424–2437, 2015.

[21] Y. Tang, L. Liu, A. Iyengar, K. Lee, and Q. Zhang, "e-ppi: Locator service in information networks with personalized privacy preservation," in *ICDCS*. IEEE Computer Society, 2014, pp. 186–197.

[22] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM*, 2001, pp. 149–160. [Online]. Available: http://doi.acm.org/10.1145/383059.383071

[23] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*, 2001, pp. 14–29. [Online]. Available: http://dx.doi.org/10.1007/3-540-45546-9_2

[24] Y. Tang and S. Zhou, "LHT: A low-maintenance indexing scheme over dhts," in *ICDCS*. IEEE Computer Society, 2008, pp. 141–151.

[25] Y. Tang, S. Zhou, and J. Xu, "LIGHT: A query-efficient yet low-maintenance indexing scheme over dhts," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 59–75, 2010.

[26] Y. Tang, J. Xu, S. Zhou, and W. Lee, "m-light: Indexing multi-dimensional data over dhts," in *ICDCS*. IEEE Computer Society, 2009, pp. 191–198.

[27] Y. Tang, J. Xu, S. Zhou, W. Lee, D. Deng, and Y. Wang, "A lightweight multidimensional index for complex queries over dhts," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2046–2054, 2011.

[28]