PRIVACY-PRESERVING COMBINATORIAL SET OPERATIONS FOR CYBER THREAT INTELLIGENCE

HARIKRISHNAN MANIKANDAN

to obtain the degree of Master of Science in Computer Science
Data Science and Technology
with a 4TU specialization in Cyber Security
to be publicly defended on 30th August 2018

DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering, Mathematics & Computer Science
Department of Intelligent Systems
Cyber Security Group

STUDENT NUMBER:
4613201

THESIS COMMITTEE:
Assoc. Prof. Dr.ir. J.C.A van der Lubbe
Assist. Prof. Dr. Z. Erkin
Assist. Prof. Dr. A. Panichella
G. Tillem, MSc

SUPERVISORS:
Assist. Prof. Dr. Z. Erkin
G. Tillem, MSc

ABSTRACT

In less than a century, the Internet has morphed from being a communication channel to a medium of existence for people. Meanwhile, attacks over the Internet have been growing both qualitatively and quantitatively, with losses transcending the financial kind and threatening the well-being of human lives. This trend fuels the need for Cyber Threat Intelligence (CTI), to change our reactive nature and start pro-actively tackling the online threats. This necessity has engendered the establishment of a number of CTI service providers. To ensure optimum expenditure, an organization would be best served by only procure intel on threats relevant to it. Hence, a scheme enabling organizations to choose the right subset of CTI service providers, to obtain the maximum amount of intel relevant to them, is needed. This scheme, however, is obliged to maintain the privacy of the inputs of all parties involved, lest the intel is leaked to the adversaries. Privacy-Preserving Set Operations offers the necessary properties and primitives to devise such a scheme. However, constructing protocols from existing solutions is infeasible, due to the exponential dependence, of their asymptotic complexities, on the number of participants in the protocol. This dissertation aims to devise protocols to enable the discussed scheme, in a feasible and privacy-preserving manner.

In this thesis, we introduce a type of composite set operations which we denote as *combinatorial set operations*. We construct four efficient, privacy-preserving protocols to compute the *set union-combinatorial intersection cardinalities* between multiple mistrusting parties, in *honest-but-curious* and *covert adversaries* models, to solve our research goal. The protocols compute the intersection cardinality between a set and all union combinations of a number of other sets, while maintaining the privacy of the set elements. All the protocols proposed in this thesis claim asymptotic complexities quadratically dependent on the number of parties involved in the protocol, while the best current alternative is constrained by an exponential dependence on the same. A comparative implementation of our principal protocol in the *covert adversaries* model boasts an overall computation time of *1* minute, for *15* parties with *5* elements in each party's set. The best current alternative requires *56.5* minutes under the same initial conditions. Moreover, even after increasing the number of participants and elements to *50* and *100*, respectively, our protocols outperformed the best current alternative subject to the previous initial constraints (*15* parties with *5* elements). We hope that the promising results obtained in this work pave the way to help organizations in navigating the increasingly complex threat environment, allowing them to pro-actively combat the rising wave of cybercrime.

ACKNOWLEDGMENT

After spending nine months on my master's thesis project, I am resigned to recognize this whole experience as a study in extremes. I now find it almost impossible to distinguish between moments when everything seemed to be crashing down and moments when I was elated with my slow, yet profoundly significant, progress. Nonetheless, facing the prospect of actually completing my thesis, I am highly appreciative of this enlightening experience and do not, in the least, regret the difficulties it entailed. Having a Bachelors degree in Electrical Engineering, I had trouble in adapting to the MSc Computer Science curriculum, which was only exacerbated when I chose a strongly academic field like cryptography to specialize in. Without the help and motivation I received from the EEMCS faculty, my peers, and my family, I would have had no other option but to quit the programme. The passion that some of the professors held, for their subjects and teaching, succeeded in engendering my interest in their topics. Their training, coupled with a lot of generous and altruistic help from my peers, and unconditional support from my family and friends literally enabled me to reach this point.

I am especially indebted to my supervisors, Zeki Erkin and Gamze Tillem. Being a novice to research, I had to depend on their vast knowledge and experience in the process, to mitigate for my erratic motivation, and wanting research acumen. I am grateful to Zeki for creating a warm environment within his research group, and interacting with us without authoritative overtones. I befriended so many of my peers thanks to Zeki's insistence, on his students, to be active in the research group. His *Q*ualified supervision, *Q*uirky demeanor, and *Q*uixotic love for chocolate cakes make for an entertaining thesis experience. I will forever cherish our enjoyably turbulent meetings, and your annoyingly valid criticisms. I am thankful, in no small parts, to Gamze for her limitless compassion, and guidance. She successfully managed to motivate and counsel me through some of the most difficult times during my thesis. It is impossible to overstate how important her composure and subject expertise were to me. I had grown so attached to her support that I faced an almost harrowing physical response to the prospect of her leaving the country. I would also like to thank Chibuike, Majid, and Oz (sometimes called Oğuzhan) for helping me with my knowledge gap in cryptography. Them, and the entire research group, have made my thesis experience so much more conducive, amusing and funky!

Finally, I thank my friends and family, who have stood by me through thick and thin. I would like to thank them, especially, for understanding my erratic behavior as I was battling stress through the thesis. Thank you all for accepting my shortcomings and standing by me through one of the most conflictingly eventful chapters of my life.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PROTOCOLS

ACRONYMS

CTI Cyber Threat Intelligence

IOC Indicator Of Compromise

ISAC Information Sharing and Analysis Centre

FI-ISAC Financial Institutions ISAC

NCSC National Cyber Security Centrum

ICT Information and Communication Technology

PPSO Privacy Preserving Set Operations

HE Homomorphic Encryption

OPRF Oblivious PseudoRandom Function

SS Secret Sharing

GC Garbled Circuit

BF Bloom Filter

OT Oblivious Transfer

TEE Trusted Execution Environment

REE Rich Execution Environment

1 INTRODUCTION

The Internet, as no other communication medium, has given a globalized dimension to the world we live in. A decisive technology of the Information Age, one can safely say that almost half the humankind is now connected to the Internet (3.8 billion as of 2017 [64]), from three in ten just seven years ago [100]. The Internet has also morphed from being just a communication channel to a medium of existence for people. In short, *"The virtual life is becoming more social than the physical life, but it is less a virtual reality than a real virtuality, facilitating real-life work and urban living"* [25].

As impressively as it has grown, the Internet is set to explode further. The number of Internet users is set to burgeon to 6 billion in 2022 and 7.5 billion in 2030 [79]. The stronger impact on the growth of the Internet, however, is not by the growth of user base but predominantly by the proliferation of smart devices that are able to interface online. The network of these smart devices that are embedded with sensors, actuators and network connectivity, while being able to function without human intervention, is called the Internet of Things (IoT) [96]. This mammoth and inexorable move is slated to connect 20.4 billion smart devices to the Internet by 2020, up from 2 billion in 2006 [77].

While the scope of Internet has undergone a paradigm shift, it has also presented a huge canvas for attacks on and through it. Securing the virtual space within the Internet, also called Cyberspace, from threats is harder than doing the same in the physical world due to the global nature of the Internet infrastructure. This predicament makes the scope of all attacks on or through the cyberspace, or Cyber attacks, potentially international and the transient nature of identification in this space makes it easier for the attackers to dissociate themselves from any possible retribution. This core problem has resulted in a rapid increase in damages incurred, which is expected to rise to 6 trillion USD in 2021 from 3 trillion in 2016 and 500 billion in 2015 [79].

Ransomwares (malware intended to hold the victim's information or privacy for ransom) infect businesses every 40 seconds in 2017 [49], which is expected to rise to a business every 14 seconds by 2019 [79]. The loss due to just ransomwares is estimated to be 5 billion dollars in 2017, up from 350 million in 2015 [80] and is growing at about 350% yearly [90]. Cyber attacks can also potentially disrupt critical infrastructures in the physical world, as evidenced by attacks on power grids [69], dams, banking systems, nuclear power plants, and water treatment facilities [11]. This disturbing trend implies that the magnitude of losses incurred due to cyber attacks has expanded to the health, well-being and survival of people.

The rapid increase in the number and loss magnitude of cyber attacks on individuals, companies and governments have prompted for a strategy to pro-

actively understand the threat environment and prepare for, rather than reacting to, attacks. Hence there is a requirement for Cyber Threat Intelligence, which will be briefly introduced in the following sub-section.

1.1 CYBER THREAT INTELLIGENCE

Threat Intelligence is an elusive concept to define, with numerous existing definitions by organizations with stakes in securing the cyberspace. According to Gartner, *"[It is] evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject's response to that menace or hazard"* [75]. In the context of cyberspace, Cyber Threat Intelligence (CTI) is a discipline that aims to assess the cyber threat environment of an organization by collecting information from relevant sources to be able to pre-empt and respond to existing or potential threats in the most effective manner. CTI also aids efficient allocation of funds for cyber security, as an accurate or even a closely approximate understanding of the threat environment to an organization helps avoid superfluous expenditure to defend against unlikely attacks.

CTI is commonly segregated, in both private and government sectors, into three levels of operation, namely: Strategic, Operational and Tactical. There are intentional overlaps in the activities performed in three levels as the intention is to frame functions and roles appropriate to the different levels rather than to establish an inflexible structure that is too rigid to meet real-world operational requirements [12]. The three levels are concisely introduced below.

STRATEGIC LEVEL The strategic level is made of the highest organizational entity of a group or an organization who are concerned with the use of the resources of the organization to realize the objectives set by them. This group of key leaders and decision makers is tasked with identifying emerging threats, framing testing scenarios and directing network defense policy. This enables senior executive decision-making on corporate strategic objectives, prioritizing cyber security/ intelligence support, and allocating resources towards the security mission with regard to the threats and other operational priorities [35].

OPERATIONAL LEVEL The operational level of cyber activities include planning, conducting and sustaining operations against hacktivists. The Chief Information Officers (CIOs) and Chief Information Security Officers (CISOs) are the key players at this level and are primarily concerned with conducting research to gather information on malicious actors. The research focus includes identifying: the adversary's technical capabilities and their development trends, the operational approach adopted by the attackers specific to the target organization, the attacker's social, legal and financial vulnerabilities and any relevant information that could potentially allow the defender to influence or outmaneuver the attacker [54].

TACTICAL LEVEL The tactical level of cyber security is concerned with the actual execution of combat elements to outmaneuver the attacker's operations upon infiltration. Most of the attention and resources of cyber defense is targeted towards this level currently. The tactical actions are typically carried out in the Network Operations Centre or Security Operations Centre. Defensive actions include: host-based security system alerts, behavior detection, and kill chain analysis based upon known attackers. While this level deserves a significant amount of attention, it should be noted that a systematic and concerted effort by the strategic and operational levels will provide the tactical level with valuable intelligence that could aid in thwarting, minimizing the damages caused by or successfully nullifying attacks [48].

1.1.1 *Cyber Kill Chain*

An exemplary illustration of an intelligence-based approach to Information Security was introduced by Lockheed Martin, popularly called the Cyber Kill Chain [74]. 'Kill Chain' was originally a military concept that explained a series of steps that had to be taken to be able to finally destroy the intended target. When adopted into the field of cyber security, the Cyber Kill Chain is a series of steps that have to be successfully taken by an adversary to be able to compromise a system successfully. This model aims to tackle a special class of threats called Advanced Persistent Threats (APTs), which represents well-trained and well-funded adversaries that conduct multi-year campaigns targeting highly sensitive economic, proprietary or national security information. A sequential flow chart of the Cyber Kill Chain is shown in Figure 1 followed by a short description of each step.

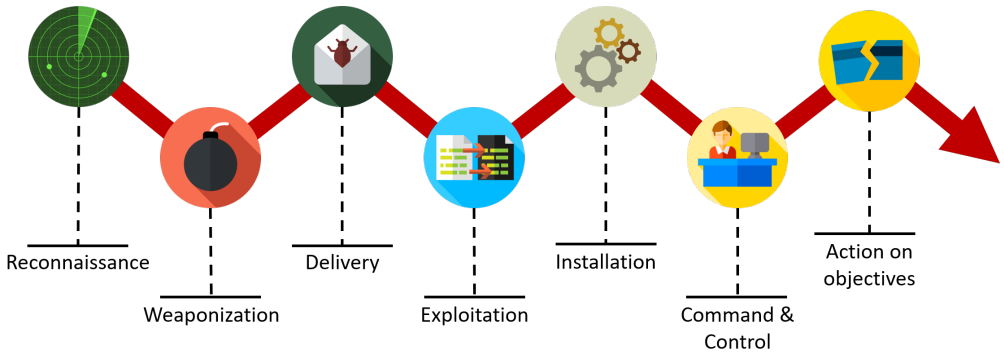


Figure 1: Cyber Kill Chain - The sequential steps taken for a successful cyber attack

RECONNAISSANCE Probe potential victims to learn about their network infrastructure, mailing addresses, social relationships and technological capabilities, often by crawling through the Internet websites relevant to them.

WEAPONIZATION Creation of malware to gain access to victims' system in the form of a deliverable payload that is designed to automatically provide

remote access to the adversary once delivered into victims' system (for example, malicious PDF or Excel documents). The malware is built using the information pertinent to the victim, collected from the previous phase.

DELIVERY Transmission of the created malware into victims' system, for example, by attaching the disguised malware created in the previous phase to an email personalized to the recipient in victims' organization (Spear Phishing).

EXPLOITATION Trigger malicious code in the malware once delivered into victims' system. This malicious code could target an application or the operating system vulnerability in victims' system.

INSTALLATION Installation of the malware, thereby gaining persistence in victims' system.

COMMAND & CONTROL Establishment of a channel to communicate with the compromised computers in victims' system to direct actions to the malware.

ACTION ON OBJECTIVES Proceed to take actions to fulfill the original objectives.

The adversary can successfully achieve his/her objective only by sequentially completing all the above steps, which implies that thwarting any single step prevents the attack from becoming successful. By accumulating information on the adversary, the detection of the attack is pushed from the latter phases to the earlier phases, thereby increasing the chances of circumventing them. Once mitigated, learning about the attacks could create an intelligence feedback loop, propelling the defenders to a state of information superiority and reducing the chances of the intrusions' success with each successive attempt [74].

1.1.2 *Indicators and Pyramid of Pain*

As evidenced in the approach explained in the Section 1.1.1, establishing and maintaining a position of information superiority requires learning from past attacks and adapting the defense to discern and prevent similar ones in the future from succeeding. Indicators Of Compromise (IOCs) play a pivotal role in this process. IOCs are signs of potentially malicious activity in a system or a network [73] that could be as simple as IP addresses associated with past malicious activities or as complex as procedures followed by adversaries during their attacks. Collecting, analyzing and correlating IOCs is not just vital to better identify security incidents but also to understand the threat environment to an organization, which is a significant part of CTI.

The level of specificity with which an IOC can be used to identify an attack or attacker greatly varies with its type. The framework that best explains the different types of IOCs, along with their utility and ease of access, is the

Pyramid of Pain [18], as shown in Figure 2 followed by a short description of the different types of indicators.

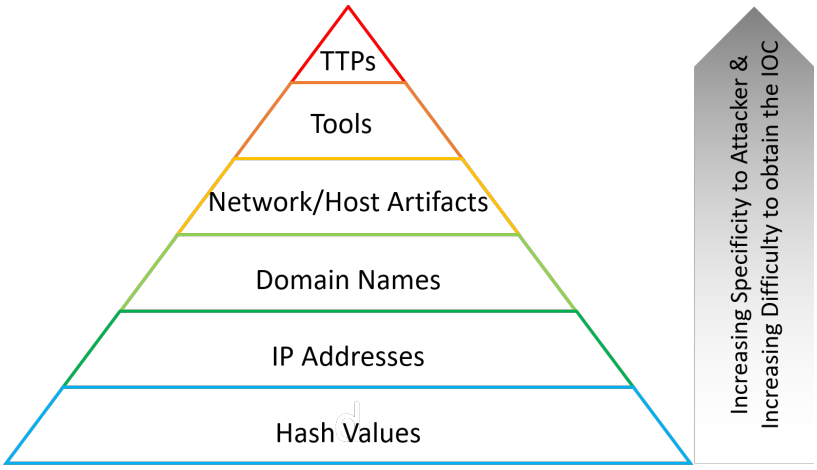


Figure 2: Pyramid of Pain[18]

HASH VALUES They are the cryptographic digests of suspicious or malicious files (For instance, using SHA3). These IOCs are very accurate as same hash values for different files are very unlikely, more so with better hashing algorithms, as even a change in just one bit would result in a completely different hash. But, this accuracy also makes it almost effortless to evade detection by just slightly modifying the file.

IP ADDRESSES As the name suggests, they are just IP addresses or blocks of IP addresses suspected to be malicious. They are fundamental indicators of cyber attacks as at least one IP address is required even to mount an attack. However, as in the previous IOC, changing the IP address after once discovered is a trivial solution and not even necessary if the adversary uses an anonymous proxy service.

DOMAIN NAMES These are domain names suspected to be involved in cyber attacks. In order for domain names to be used, they have to be registered, paid for and hosted somewhere, which could take one to two days before they are up and can be used. This makes it harder for the adversary to change them upon detection as it incurs costs to create new domain names and the delay in doing so. However, a large number of Domain Naming System (DNS) providers mitigate the inconvenience partly.

NETWORK/HOST ARTIFACTS Network Artifacts are observables that are caused by the adversary's activities in the defending organization's network. This could be any deviation from normal network behavior, like additional load due to the command & control channel established with the victim's computer or due to denial of service attacks on the victim's system. Host Artifacts, similarly, are observables noticed in a defending organizations host(s). These could be of any form such as malware, suspicious registry entries or files that are indicative

of adversarial actions. These IOCs are harder to deal with than all the ones before as detecting artifacts and solving the breach would force the adversary to modify the functioning of the malware so as to not leave the same artifacts as before.

TOOLS These are software used by the adversary in the process of their attack. They are likely to be brought in by the adversary to achieve their goal and less so that they are already installed in the victim's computer, like tools required to establish command & control. Identifying the tools that are specific to the attacks precludes the possibility of the tool being used in a successful attack again, forcing the adversary to use different tools for the same purpose which requires time for research, development and training.

TACTICS, TECHNIQUES & PROCEDURES These are methods used by the adversary to accomplish his/her goals, anything from reconnaissance to action on objectives or all of them. They could be as simple as simply spear phishing to as specific as the entire process of the attack. Once identified by the defenders, it is very difficult for the attacker to successfully attack the defender as this requires a fundamental change in the behavior of the attacker.

As easily observed from the above description of the IOCs, the indicators become more specific to the attacker as it goes higher up in the pyramid and also harder for the attacker to recover from. However, obtaining the IOCs also becomes more difficult as it goes higher in the pyramid [18].

1.1.3 *Collaborating Threat Intelligence*

The vast number of attacks, which also continues to grow rapidly, makes it a difficult challenge for any single organization to accrue the adequate amount of relevant information to develop accurate situational awareness of the threat landscape [14]. The problem is also exacerbated by the fact that there are possible communication and collaboration between adversary communities from different countries, as observed in an analysis on the tools utilized by the adversaries to probe their prospective victims [28]. One solution to overcome this information gap is by subscribing to Threat Intelligence service providers. Services by LookingGlass[1], SecureWorks Inc.[2], FireEye Threat Intelligence[3], etc., that can be bought, can help to provide information pertinent to organizations on even advanced threats, such as, zero-day attacks and APTs [78].

Threat information-sharing is another solution to mitigate the problem. A number of information-sharing platforms such as AlienVault's Open Threat

1 https://www.lookingglasscyber.com/
2 https://www.secureworks.com/
3 https://www.fireeye.com/solutions/cyber-threat-intelligence.html

Exchange[4], and Critical Stack's Intel[5] exist for this exact purpose. Uniform standards for collecting and communicating threat intelligence information like STIX[6], TAXII[7] and CybOX[8] are also being developed to facilitate the information-sharing process. To make information sharing within the individual sectors conducive, Information Sharing and Analysis Centres (ISACs) have been formed for the individual sectors. The ISACs have representatives from several organizations belonging to its particular sector and also the appropriate government bodies, and meet periodically to exchange information & experiences regarding cyber security [59]. The National Cyber Security Centrum (NCSC) of the Netherlands involves itself with ISACs from several sectors, viz. port, airport, nuclear, water management etc [59]. ISACs are also established at continental levels, such as the European Financial Institutions ISAC (FI-ISAC) which aims to facilitate information exchange on all topics pertinent to Information and Communication Technology (ICT) of the financial community, including cyber criminal activities, vulnerabilities, technology trends, threats, incidents and case-studies [84].

However, there could be substantial costs to an organization in sharing such information including and not limited to reputation damage, legal consequences, loss of client information to competitors and leakage of critical information to the adversaries. Use of Trusted Third Parties is one solution to this problem, but that does not preclude the possibility that trusted parties themselves could be compromised, especially considering that they are attractive targets [62]. Hence, necessitating privacy-preserving solutions to threat information-sharing. This necessity has prompted a number of research works in this area [3, 62]. Since the information sharing requires privacy-preserving solutions to compare unordered lists of information, held by various parties, privacy-preserving set operations offer the necessary primitives to construct the information-sharing platform.

1.2 PRIVACY-PRESERVING SET OPERATIONS

Privacy-Preserving Set Operations (PPSO) can be used to devise solutions (as mentioned in Section 1.1.3) for threat information-sharing, especially when comparing sets of information between multiple parties. The intended objective of the operations is to maintain the confidentiality of the input lists (or sets) from other parties while the result of the comparative computations is available to a selected few or all the parties. The levels of privacy involved could vary between just keeping the elements of the input set private to also keeping the number of elements in a set private. The different assumptions of privacy will be elaborated on in Chapter 2. A popular example for an

4 https://otx.alienvault.com/
5 https://intel.criticalstack.com/
6 https://stixproject.github.io/
7 https://taxiiproject.github.io/
8 https://cyboxproject.github.io/

application of PPSO is that of comparison between airline passenger lists with national watch lists [7, 19, 67, 95]. Airlines are required to share their passenger list to the US Department of Homeland Security, to check if there are any common names between the flight passenger list and the Terror Watch List (a dynamic database of suspected terrorists). The scenario requires the airlines to keep its customer information private, while the Department of Homeland Security would find it imperative to keep its list private too. Hence, this requires the computation of set intersection between the two lists in a privacy-preserving environment.

Research works in this field have focused predominantly on basic operations like set intersection, set union and set cardinality between two or more parties. There is also strong importance placed in the reduction of computation and communication costs of these protocols, so as to make them practical to implement. This importance makes PPSO based solutions ideal for threat information-sharing. However, while the three basic operations could be used as building blocks for more advanced applications, it could result in high communicational and computational complexities due to inefficiently running the basic operations sequentially and repetitively [19]. This dissertation aims to provide an efficient solution for a non-basic application, which will be detailed in the following section.

1.3 RESEARCH GOAL

As established before, the threats to the cyber security of an organization cannot be addressed by simply improving its defenses to prepare for previously faced attacks. The pressing need to identify and fix vulnerabilities in its system by proactively studying its threat landscape before an attack even takes place is now more than ever before. This need has given rise to a number of CTI service providers. However, the knowledge of which service provider or a group of them is fruitful to an organization is difficult to deduce without purchasing their services first.

Instead, if the organization has some information on the threat environment in the form of low-level IOCs, viz. IP addresses or hash values, it can purchase higher level IOCs, viz. Network/Host Artifacts, Tools and Techniques, Tactics & Procedures, that correspond to the lower level IOCs. The low-level IOCs could be collected during the earlier stages of the Cyber Kill Chain of the adversary. The above solution requires the organization to devise a way to determine a subset of the available service providers that best covers its list of low-level IOCs.

This scenario can also be generalized to quantitatively identifying the appropriate subset of vendors by a client, based on her requirement of them. This is especially so when the vendors offer subscription-based services, where the clients pay for all items entailed in that service instead of per item. Subscription based services, such as Netflix and Spotify, are already burgeoning and increasingly replacing their traditional counterparts as the

dominant players in their respective fields [27]. With the proliferation of such subscription-based services, their advent into fields which require confidentiality of what is entailed in a vendor's service (such as in Cyber Threat Intelligence) has to be facilitated by devising systems that enable clients to choose their appropriate vendors, while still maintaining the privacy of clients and vendors alike. The goal of this dissertation is derived from solving the above problem, and it can be succinctly framed as below.

> *"How to devise a protocol to enable a party (Client) with a set of items, chosen from a finite universe, to find a subset of other parties (Vendors) with similar sets, in such a way that the union set, of all vendors in the said subset, has the maximum set intersection cardinality with the Client's set, while maintaining the privacy of all participants' set elements?"*

Let us consider the scenario where there is a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ such that P_1 is the *client*, and the rest are *vendors* belonging to the set $\mathcal{V} = \{P_2, \ldots, P_n\}$. Each party P_i, $i \in [1, n]$ holds the set S_i with k_i elements drawn from a finite universe $\mathcal{U}$. Additionally, let $\Delta\,(= \binom{\mathcal{V}}{1} \cup \ldots \cup \binom{\mathcal{V}}{n-1})$ represent the set of all non-null combinations of *vendors* and S_δ represents the union of the sets held by all the parties in the combination $\delta \in \Delta$. Our research question requires evaluating $|S_1 \cap S_\delta|$, $\forall\, \delta \in \Delta$. We denote this operation as the *set union-combinatorial intersection cardinality*, and the family of such operations as *combinatorial set operations*.

The research question can be divided into the following sub-questions:

1. How to ensure that no privacy-leaking intermediate transcripts are shared during the protocol?
2. How to ensure that any collusion, between the *client* and an arbitrary subset of *vendors*, does not leak any additional information about the involved parties' sets other than the result of the protocol?
3. How can the social circumstances of the involved parties be leveraged to construct an efficient protocol?
4. Will the constructed protocol be practically more efficient than the most efficient current alternative?

1.4 OUR CONTRIBUTIONS

In this work, we build four protocols to compute our desired result,

1. Private set union-combinatorial intersection cardinality in *honest-but-curious adversaries* model (SUCCINCT-HBC),
2. Private set union-combinatorial intersection cardinality in *covert adversaries* model (SUCCINCT),

3. SUCCINCT with output obfuscation (SUCCINCT-O),

4. SUCCINCT with d-out-of-n security (SUCCINCT-d),

to compute the set union-combinatorial intersection cardinality. These protocols do not compute the intersection cardinalities itself, and instead results in a data structure, $\mathcal{M}$. The *client* can then compute the intersection cardinality between her set and the union of sets held by any subset of *vendors* using $\mathcal{M}$, without having to interact with any other party. The result format is chosen so, as for $n-1$ *vendors* there are $2^{n-1}-1$ combinations of them. This predicament implies that any approach, devised to compute the intersection cardinality with every combination's union, has a computation complexity of at least $\mathcal{O}(2^n)$. To mitigate this issue, we advocate using optimizations to obtain the appropriate subset of *vendors* from $\mathcal{M}$. An optimization algorithm, cOMbO, to identify a smallest subset of *vendors*, s.t. the union set of this subset has the maximum intersection cardinality with the *client's* set among all possible subsets, is also presented.

SUCCINCT-HBC, SUCCINCT, SUCCINCT-O and SUCCINCT-d are all based on modular exponentiations and Bloom filters, which ensure the privacy of the input set elements. In addition to them, we have also built two additional preliminary protocols,

1. Secure simple multi-party coin tossing in *malicious adversaries* model (SCULPT)

2. Secure shuffle and exponentiation in *covert adversaries* model (SESAmE).

Our contributions are as follows:

- To the best of our knowledge, we present the first protocols that compute *union-combinatorial intersection cardinality* in *honest-but-curious adversaries* and *covert adversaries*. Both protocols boast a computational complexity of $\mathcal{O}(n^2k - nk\ln(\epsilon))$ and communicational complexity of $\mathcal{O}(n^2k(1-\ln(\epsilon)))$, for n participants with k elements in each participant's set, and ϵ being the false positive rate of the employed Bloom filters. The alternatives require at least $\mathcal{O}(2^n)$ computation and communication complexity to compute the same. A naive implementation of SUCCINCT requires only *1* minute of computation time for *15* parties with *5* elements in each party's set, while the most efficient current alternative takes *56.5* minutes.

- We propose a protocol for shuffling and exponentiation of a set by a party, secure in *covert adversaries*, with $\mathcal{O}(k)$ asymptotic complexities.

- Our first modification, SUCCINCT-O, improves on the security of SUCCINCT, by making the protocol robust to collusion between malicious *client* and malicious *vendors*, without affecting the asymptotic complexities of the protocol.

- Our second modification, SUCCINCT-d, reduces the computation complexity of SUCCINCT, by reasonably compromising on the security of the joint secret key, without relaxing the security model of the protocol.

SUCCINCT-d requires $\mathcal{O}(ndk + dk\ln(\epsilon) + \frac{n^2}{d})$ computation complexity, and $\mathcal{O}(ndk(1 - \ln\epsilon) + n)$ communication complexity, both of which are improvements on the asymptotic complexities of SUCCINCT, given that d is always lesser than n.

- Our optimization algorithm, cOMbO, presents an efficient way to obtain our required subset of *vendors* from the results of SUCCINCT-HBC, SUCCINCT, SUCCINCT-O, and SUCCINCT-d. cOMbO identifies the smallest subset of *vendors* whose union-combined set has the maximum possible overlap with the *client's* set among all subset combinations of *vendors*. The protocol requires maximum $\mathcal{O}(n^2k)$ plain-text operations, while just the number of different possible combinations of *vendors* is $2^{n-1} - 1$.

A flow chart explaining the functional flow of the protocols constructed in this thesis is provided in Figure 3. SUCCINCT-HBC, SUCCINCT, SUCCINCT-O, SUCCINCT-d are all a family of protocols that produce the same result, which is the input to cOMbO, the output of which is the desired result of this thesis. The preliminary protocols SCULPT and SESAmE, along with SUCCINCT-HBC, are used to construct our principal protocol SUCCINCT.

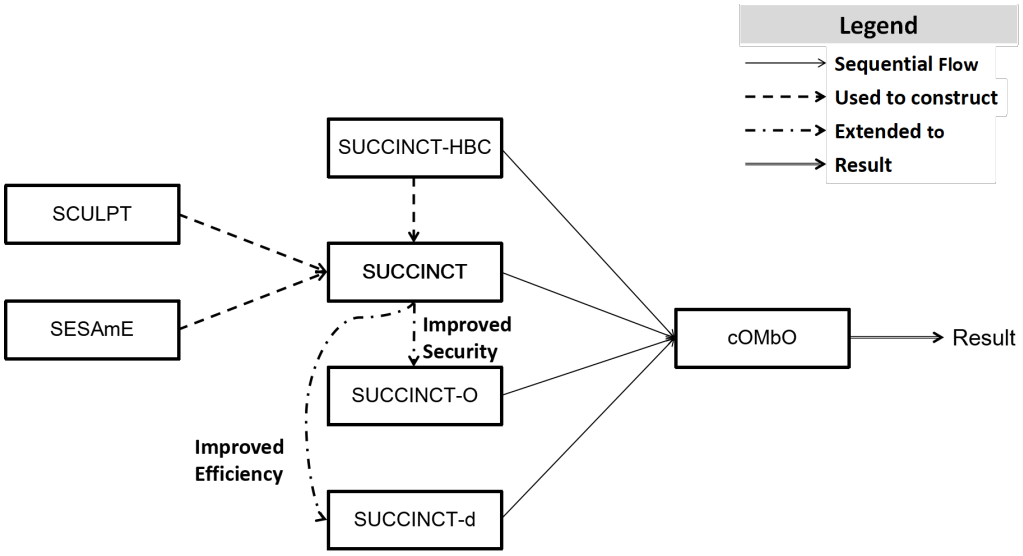


Figure 3: Protocol Functional Flow

1.5 THESIS OUTLINE

We structure our thesis to the following outline. Chapter 1 introduces CTI, the need for it, and motivates our research question. Chapter 2 provides a brief summary of the earlier works in the field of PPSO along with the necessary primitives to understand them, with an emphasis on the diversity of techniques employed. Chapter 3 exhibits the preliminary protocols used to construct our principal protocol, along with their complexity and security analyses. Chapter 4 details the specifications and analyses of our principal protocol. Chapter 5 proposes some modifications to our principal protocol.

Chapter 6 explains how to optimally obtain the required result. Chapter 7 compares the performance of our protocols with the most efficient current alternative, and finally, Chapter 8 discusses the obtained results and suggests some future work to follow the research work presented in this dissertation.

2

PRIOR ART

The protocols for PPSO are built on assumptions regarding the capability and intent of plausible adversaries involved in them, and the level of privacy required of the protocol. The adversarial model assumed in a protocol defines the expected actions of adversaries, if any, and how robust the protocol is to adversary attacks. In the following paragraphs, we discuss the adversarial models that are commonly assumed in the research works in this field.

HONEST-BUT-CURIOUS (OR SEMI-HONEST ADVERSARY) In this model, the participants of the protocol are assumed to follow the protocol specifications accurately. However, they will try to learn as much as possible from the protocol iterations and examine its messages more than they are supposed to [42].

MALICIOUS ADVERSARY The adversaries in this assumption are assumed to deviate from the protocol specification as arbitrarily as required by their attack strategies. This deviation could be to derive more information from other parties, alter the protocol result and/or terminate it prematurely [42].

COVERT ADVERSARY *Covert adversaries* are between *honest-but-curious* and *malicious adversaries*. The motivation behind this assumption is that in real-world settings there exist adversaries that are willing to cheat the protocol only as long as they are not caught. This constraint maybe due to unaffordable losses such as embarrassment, loss of reputation and negative press [52]. Hence, the model is required to constrain the probability with which the adversary can successfully cheat without getting caught.

ONE-SIDED SIMULATABILITY One-sided simulatability is a special case adversarial model where the protocol involves only two participants and only one of them receives the output of the protocol. Proponents of this model aim to maintain the privacy of both the parties inputs [52]. Due to the asymmetric nature of this model, only privacy by indistinguishability is required from the viewpoint of the participant who receives the output. Hence, the actions of the other participant can affect the protocol execution but that will not be sufficient to compromise the privacy of the protocol and its inputs.

The assumed adversarial model strongly influences the computational and communicational complexities of the protocol. Improving the privacy of a protocol is naturally likely to decrease its efficiency. *Covert adversaries* and *one-sided simulatability* models were, in fact, conceptualized to provide better privacy than the simple *honest-but-curious adversaries* model and yet be feasibly efficient enough to be implementable, as compared to the *malicious adversaries* model [51]. While, the works provide solutions for a number of different set

operations, our focus will be limited to set intersection, set union and set cardinality protocols. The following sections provide a review of the prior work performed in the field of PPSO. They segregate the works based on the primary component of the methodology employed, and presents the required primitives to understand them, if any. The final two sections provide a comparative analysis of the summarized solutions, and the open issues that still remain in solving our problem.

2.1 HOMOMORPHIC ENCRYPTION BASED SOLUTIONS

2.1.1 *Primitives*

HOMOMORPHIC ENCRYPTION Homomorphic Encryption enables the execution of one, a few, or all basic mathematical operations on encrypted values. By doing so, these operations can be executed without revealing the plaintext values, while still obtaining the intended result in its encrypted form. In the field of PPSO, the most common form of homomorphic encryption used is the additive variant.

Additive Homomorphic Encryption Additive Homomorphic Encryption is a subset of Homomorphic Encryption schemes, which only facilitates addition and subtraction of plaintexts by executing preset operations on two or more encrypted values [42]. For instance, supposing $\mathcal{E}_k(m_1)$ and $\mathcal{E}_k(m_2)$ are the encrypted values of plaintexts m_1 and m_2 respectively, there exists an operator $\star$ such that,

$$\mathcal{E}_k(m_1) \star \mathcal{E}_k(m_2) = \mathcal{E}_k(m_1 + m_2) \tag{1}$$

The nature of the operator $\star$ depends on the type of encryption scheme.

Paillier Cryptosystem Paillier is a probabilistic asymmetric key cryptosystem, that is also additively homomorphic [86]. The concept and the different phases of the system are apprised below.

- **Concept -** Let $n = pq$, where p and q are large primes, $\phi(n) = (p-1)(q-1)$ and $\lambda = lcm((p-1),(q-1))$. Then, for any $w \in \mathbb{Z}^*_{n^2}$,

$$\begin{aligned} w^{\lambda} &= 1 \, mod \, n \\ w^{\lambda n} &= 1 \, mod \, n^2, \end{aligned} \tag{2}$$

  in accordance with Carmichael's Theorem.

- **Setup -** Following the notation format from the previous point, let $g \in \mathbb{Z}^*_{n^2}$. Then, the public key $pk = (g, n)$ and secret key $sk = (\lambda, \mu)$, where
  $\mu = (L(g^{\lambda} \, mod \, n^2))^{-1} \, mod \, n$.

- **Encryption -** For a message $m \in \mathbb{Z}_n$, the encryption formula is,

$$\mathcal{E}_{pk}(m) = g^m r^n \, mod \, n^2, \tag{3}$$

where $r \in_R \mathbb{Z}_n^*$.

- **Decryption -** For ciphertext $c = \mathcal{E}_{pk}(m)$ of the plaintext m, the decryption formula is,

$$\mathcal{D}_{sk}(c) = L(c^{\lambda} \bmod n^2) \times \mu \bmod n = m \tag{4}$$

- **Homomorphism -** For two ciphertexts $c = \mathcal{E}_{pk}(m)$ and $c' = \mathcal{E}_{pk}(m')$, they are additively homomorphic in such a way that,

$$\begin{aligned} c \times c' = \mathcal{E}_{pk}(m) \times \mathcal{E}_{pk}(m') &= g^m r^n \bmod n^2 \times g^{m'} r'^n \bmod n^2 \\ &= g^{m+m'} r^n r'^n \bmod n^2 = \mathcal{E}_{pk}(m + m') \end{aligned} \tag{5}$$

POLYNOMIAL REPRESENTATION OF SETS A multi-set, $S = \{s_1, ..., s_n\}$, can be represented as a polynomial function, $f_S(x) = \prod_{i=1}^{n}(x - s_i)$. This function f_S is a polynomial representation of the multi-set S. A vital property of this representation is that for any element $y \in S$, $f_S(y) = 0$ [42]. Moreover, hiding the multi-set S just requires the encryption of the coefficients of the polynomial f_S. Suppose the polynomial is of the form $f_S(x) = \sum_{i=0}^{n} a_i x^i$,

$$\mathcal{E}_K(f_S) \implies \{\mathcal{E}_K(a_n), ..., \mathcal{E}_K(a_0)\} \tag{6}$$

Encrypting the coefficients using an additively homomorphic encryption scheme, such as Paillier [86], allows for a number of operations to be performed on the encrypted polynomial, as shown below:

- **Polynomial Evaluation -** Given a known value y, $\mathcal{E}_K(f_S(y))$ can be easily computed using the formula

$$\mathcal{E}_K(f_S(y)) = \prod_{i=0}^{n} \mathcal{E}_K(a_i)^{y^i} = \mathcal{E}_K(\sum_{i=0}^{n} a_i y^i) \tag{7}$$

- **Polynomial Addition -** Given two polynomial functions, f and g, it is possible to compute $\mathcal{E}_K(f + g)$ from $\mathcal{E}_K(f)$ and $\mathcal{E}_K(g)$, using the additive homomorphic property.

COMMITMENT SCHEMES Commitment Schemes enables a sender to commit to a secret value, without divulging it, and also restraints them from changing the secret value before revealing the same [99]. This is done by creating a "commitment" using the secret value in such a way that it is infeasible to both learn the secret value from the commitment (Hiding Property), and alter the secret value while keeping the commitment the same (Binding Property).

Pedersen Commitment Scheme Pedersen commitment is a type of commitment scheme that allows for information-theoretical hiding of a secret and computational binding to the same [99]. The structure of the commitment, and its intrinsic properties are succinctly introduced as follows,

- **Structure -** Let g, h be generators of the multiplicative group G with prime order q. Then, the Pedersen commitment of a secret message z is structured of the form $PC_r(z) = h^z g^r$, where $r \in_R \mathbb{Z}_q$. In the revealing stage the values of z and r are shared with the other parties so that they can verify that the choice was not altered prior to revealing.

- **Binding Property -** Let there exist two sets of values *(z,r)* and *(z',r')*, such that $PC_r(z) = C_{r'}(z')$. This implies that

$$r' = r + (z - z')t, \ \textit{where } t = \log_g(h) \tag{8}$$

  Now, any Pederson commitment over z can be modified to z' by computing the appropriate value of r' using the Equation 8, but only as long as the value of t can be calculated. However, solving for t is a Discrete Logarithm Problem and is computationally not feasible [99]. Hence, Pedersen Commitments are only computationally binding.

- **Hiding Property -** Creating a Pedersen Commitment $PC_r(z)$ requires generating an $r \in_R \mathbb{Z}_q$. Naturally, g^r is also random as a result. Hence, $PC_r(z) = h^z g^r$ is random, as the product of a random number and any non-zero value is still random. Hence, Pedersen commitments are information-theo- retically hiding as there is no way to eliminate the random value in it without revealing it.

ZERO-KNOWLEDGE PROOFS It is often required of a party to prove that he/she possesses the knowledge of a secret value and/or has correctly computed a function using secret values previously committed to, without revealing the secret value(s). Protocols that enable such proofs are called Zero-Knowledge Proofs, as the knowledge of no secret value is betrayed to the other parties involved [99]. An example of an interactive Zero Knowledge Proof of Knowledge using a Pedersen Commitment is shown in Protocol 1. The aim of the protocol is for Alice to prove to Bob that she knows the secret message z that is used to construct the Pedersen Commitment $PC_r(z)$. Let $g \in \mathbb{Z}_q^*$ be a generator of the multiplicative group G of prime modulo q. All operations are computed in mod q.

Such interactive protocols, where zero-knowledge proofs are used to prove knowledge, are called Σ-protocols. They are called so since, once the statement to be proved is established, the subsequent steps are the same. *1)* The prover (Alice) creates what is called a Commitment, *Co* (this is not to be mistaken with the ones introduced in 2.1.1), and transmits it to the verifier (Bob). *2)* The verifier responds with a Challenge, *Ch. 3)* Finally, the prover transmits a Response, *RE*, that is used to make the final verification. These interactive proofs can also be made non-interactive by employing the heuristics suggested by Fiat and Shamir [40]. This way, an interactive proof can be modified to a digital signature and reduce the communication overhead.

Protocol 1 : Zero-knowledge Proof of Knowledge

Alice		Bob
z, r, $y = PC_r(z) = h^z g^r$		$y = PC_r(z)$
k_1, $k_2 \in_R \mathbb{Z}_q$		
$Co = h^{k_1} g^{k_2}$	$\xrightarrow{Co}$	
	$\xleftarrow{Ch}$	$Ch \in_R \mathbb{Z}_q$
$s_1 = k_1 + z \times Ch$		
$s_2 = k_2 + r \times Ch$	$\xrightarrow{s_1, s_2}$	
		Verify: $h^{s_1} g^{s_2} = y^{Ch} \times Co$

2.1.2 *Existing Literature*

The two-party solutions for Set Intersection and Intersection Cardinality by Freedman et al. [41] were the first custom solutions in the field of PPSO. They proposed solutions, using the additively homomorphic Paillier system [86] and polynomial representation of sets, to securely compute the Intersection and Intersection Cardinality between a Client's and a Server's private lists, in the *honest-but-curious adversaries* model. The protocol, secure in *honest-but-curious adversaries* model, utilizes polynomial evaluations in the encrypted domain with a communication and computation complexity of $O(k)$ and $O(k^2)$ respectively, where k is the size of both parties' input sets. They, however, also suggest optimizations using balanced hash functions, such as in [10], to reduce the overall computational complexity to $O(k(\ln \ln k))$. The paper also proposes variants to the above protocol for the cases when one of the parties is honest and the other is malicious, along with a sketch on how to construct a protocol secure when either or both can be malicious. Zero-knowledge proofs are avoided in the malicious variant of the protocol to prevent high computation complexities. However, they suggest cut-and-choose method in one of the variants, which could inflate the communication overhead. Also, the work does not use any other property of polynomials apart from its evaluation at given points.

Kissner and Song [67] extended the work of Freedman et al. by proposing solutions for secure set and multi-set operations including set union, set intersection, and element reduction. Moreover, their work was the first to establish protocols which were secure under the *malicious adversaries* model when three or more parties were engaged. The idea behind their protocols is that given the polynomial representations of two sets, f_X and f_Y (of sets X and Y respectively), the roots of the polynomial $f_X \times r + f_Y \times s$ coincides with the multi-set $X \cap Y$ with high probability, where r and s are polynomials chosen at random. Similarly, the roots of $f_X \times f_Y \times r$ coincides with $X \cup Y$ with high probability. The protocols are constructed with a *(n,n)* threshold variant of Paillier cryptosystem [86], such that decryption is only possible when all

parties do it jointly. The work also establishes security in the *malicious adversaries* model by incorporating zero-knowledge proofs to validate all parties' adherence to the protocol. This has a detrimental impact on the communication and computation overhead of the protocols. Also, the set representing polynomials are randomized using random polynomials, implying that the randomization process polynomial multiplication in the encrypted form, which incurs a severe computation overhead. However, utilizing the polynomial properties that are suggested in this work, it is possible to construct complex set operations by computing the appropriate composite polynomial.

Frikken [42] proposed efficient solutions for Set Union and Union Cardinality between multiple parties with a reduced communication complexity of $O(n^2k^2 + n^3k)$ in the *honest-but-curious adversaries* model, for n parties with k elements each. The work utilizes the concept of constructing composite polynomials to obtain the union of sets and a *(n,n)* threshold variant of Paillier cryptosystem [86] to encrypt the polynomials, similar to [67]. However, instead of decrypting the composite polynomials, all parties jointly decrypt tuples of polynomial evaluations. Hence, the randomization of the tuples is done by multiplying them with random numbers instead of polynomials, which involved a lesser computational overhead than that in [67]. The protocols are made secure in the *malicious adversaries* model using zero-Knowledge Proofs, similar to [67] again. The paper also suggests a method to obscure the magnitude of a party's set by adding an arbitrary number of the element zero to his/her set, but this requires the element zero to be excluded from the domain of the set elements. Also, their solution to empty-set attacks acknowledge that although there are countermeasures that can be taken, there is no general solution to this problem.

Hazay and Lindell [50] suggested a protocol for efficient Oblivious Polynomial Evaluations, secure under the *malicious adversaries* model. The proposed protocol can be easily modified into a two-party set intersection protocol, similar to OPRFs, using the framework suggested in [51]. Each oblivious transfer protocol has a computation complexity of $O(ds)$, where d is the degree of the secret polynomial and s is the security parameter (equal to 160 in practice [50]). The security in the protocol is established using zero-knowledge proofs to validate each party's intermediate output, and cut-and-choose in the final step to prove that a vast majority of the of the computations are correct. A disadvantage of this protocol is that it requires generating and performing encrypted evaluations on s random polynomials of degree d to randomize the original polynomials. Moreover, the encryption scheme used in the protocol is Paillier [86], which implies that the exponentiations required for the polynomial evaluations are done in a high modular space. The computation complexity is adversely affected as a result.

Hazay and Nissim [53] furthered the work of [41] by establishing two-party set intersection and set union protocols, secure in the *malicious adversaries*

model. The protocols are based on encrypted polynomial evaluations utilizing the additive variant of El Gamal encryption scheme [44] and balanced allocations [10], similar to [41]. The protocol is made secure in the *malicious adversaries* model using zero-knowledge proofs, perfectly binding Pedersen Commitments [89] and an oblivious pseudorandom function (such as [82]. The protocols for set intersection and union, secure in the *malicious adversaries* model, have almost linear computation complexities $O(k(\log\log k + l))$ and $O(kl)$ respectively, where k is the number of elements in each party's set and l is the number of bits used to represent a set element. An interesting aspect of the proposed solution is that the use of zero-knowledge proofs are only required to prove the correct computation of polynomials and to prove knowledge of the initial secret key. The expensive process of proving the correct evaluation of the polynomials is not required, positively affecting the computation overhead.

2.2 Oblivious pseudorandom and unpredictable function based solutions

2.2.1 *Primitives*

1-out-of-2 oblivious transfer 1-out-of-2 Oblivious Transfer (OT) protocols allows a recipient to choose and receive one of two messages from a sender, without the sender learning the receiver's choice and the receiver learning the value of the unchosen message [99]. An example protocol for 1-out-of-2 OT is shown in Protocol 2. This protocol is based on the solution suggested by Even et al. [39], and instantiated using RSA encryption system [93]. Let n be an RSA modulo with public key e and private key d.

Protocol 2 : 1-out-of-2 OT

Sender		Receiver
$z_0, z_1;\ pk = (e,n);\ sk = d;\ n = pq$		$b \in \{0,1\};\ pk = (e,n)$
$x_0, x_1 \in_R \mathbb{Z}_n$	$\xrightarrow{x_0, x_1}$	
		$r \in_R \mathbb{Z}_n$
	$\xleftarrow{v}$	$v = (x_b + r^e)$
$z'_0 = ((v - x_0)^d \mod n) + z_0$		
$z'_1 = ((v - x_1)^d \mod n) + z_1$	$\xrightarrow{z'_0, z'_1}$	
		$z'_b = z_b - r$

oblivious pseudorandom functions Let there be two mutually mistrusting parties, Alice with a secret message z, and Bob with a secret key K and a Pseudorandom Function $\mathcal{F}_{PRF}$. Oblivious Pseudorandom Functions

enables Alice to obtain the evaluation of the Pseudorandom Function (which takes both the secret message z and key K as input; i.e. $\mathcal{F}_{PRF}(z, K)$), with neither Alice nor Bob ascertaining the other party's input(s) [51]. One important advantage of OPRFs is that the function evaluations for multiple elements can all be done in parallel, as each evaluation is independent of the other elements in a set.

Naor-Reingold OPRF Hazay and Lindell [51] suggested an OPRF protocol based on obliviously evaluating the Naor-Reingold PRF [82] without the knowledge of its secret key. An example evaluation of Naor-Reingold based OPRF is shown in Protocol 3. Let the two participants of the protocol be P_1 and P_2. P_1's input to the protocol is the secret key $k = \{p, q, g, g^{a_0}, a_1, ..., a_\ell\}$, where p is a prime order of a group G, q is a l-bit prime divisor of p, $g \in \mathbb{Z}_p^*$ is of order q and $a_0, ..., a_\ell \in_R \mathbb{Z}_q^*$. P_2's input is his/her ℓ-bit secret element $x = (x_1, ..., x_\ell)$, where $x \in \{0,1\}^\ell$ and $x_i \in \{0,1\}$, $i = 1, ..., \ell$.

Protocol 3 : Naor-Reingold based OPRF evaluation

P_1		P_2
$k = \{p, q, g, g^{a_0}, a_1, ..., a_\ell\}$		$x = (x_1, ..., x_\ell)$
$r_1, ..., r_\ell \in_R \mathbb{Z}_q^*$		
$(r_i, r_i \cdot a_i),\ i = 1, ..., \ell \rightarrow$	$\xleftrightarrow{\ell\ OTs}$	$\leftarrow x_i,\ i = 1, ..., \ell$
	$\longrightarrow$	$y_i = r_i(a_i)^{x_i},\ i = 1, ..., \ell$
$\tilde{g} = g^{a_0 \cdot \prod_{i=1}^{n} \frac{1}{r_i}}$	$\xrightarrow{\tilde{g}}$	
		$\mathcal{F}_{PRF}(x, k) = y = \tilde{g}^{\prod_{i=1}^{n} y_i}$
		$= g^{a_0 \cdot \prod_{i=1}^{n} a_i^{x_i}}$

2.2.2 *Existing Literature*

Hazay and Lindell [51] were the first to propose two-party set intersection protocols build with Oblivious Pseudorandom Functions (OPRFs). The protocols are built for two adversary models, viz. *one-way simulatability* and *covert adversaries* model. The relaxed security constraints of the two models enable them to build protocols that are efficient and yet provide adequate privacy for practical requirements. The protocol secure in one-way simulatability model is constructed using 1-out-of-2 OT [2, 81] and Naor-Reingold pseudorandom function [82]. The idea behind protocol is that the party with neither the secret key nor the pseudorandom function obtains the pseudorandom function evaluation for all the elements in both parties' sets. The intersection set is derived by comparing the evaluated lists of both parties. The advantage of this technique is that the privacy of both parties' inputs is preserved without the use of any computationally expensive zero-knowledge proofs. However, there is no way to verify the correctness of the protocol. The

protocol secure in *covert adversaries* model is of a more complex construct. In addition to the tools used in the previous one, this protocol uses Pseudorandom Permutation functions, a perfectly binding commitment scheme, a coin-tossing protocol [70] and an OT protocol secure in the *covert adversaries* model with deterrent factor $^1/_2$. The deterrent factor of the protocol is also $^1/_2$, implying that an adversary is allowed to cheat but only with a success probability of $^1/_2$. Both the protocols in this paper has a communication and computation complexity of $O(kl)$, where l is the number of bits used to represent the input elements, and k is the set size of each party. A drawback of the two suggested protocols is that obtaining the pseudorandom function evaluation for an element requires *O(l)* OTs which adversely affects their communication overheads.

Jarecki and Liu [60] improved the work presented in [51] by constructing the two-party set intersection protocol, secure in *malicious adversaries* model, using a pseudorandom function accompanied with a committed key. The commitment ensures that the same key is used through the protocol. They propose a committed OPRF using Camenish-Shoup version [23] of Paillier cryptosystem, which could easily be converted into a set intersection protocol using the framework from [51]. The security in the protocol is established using efficient zero-knowledge proofs that can be achieved with *O(1)* computation complexity, descriptions of which are given in [60]. The work boasts a linear computation complexity of *O(k)*, where k is the maximum size of both sets, in spite of achieving security in *malicious adversaries* model.

Efficient solutions for two-party set intersection were proposed by Cristofaro et al. [30]. These solutions were secure under the *malicious adversaries* model with linear communication and computation complexity. One of the protocols, termed Authorized Private Set Intersection, constrained one of the parties to use signed set elements as inputs. They suggest using RSA Cryptosystem [93] for obtaining the digital signatures from an authorized entity. The other protocol is a regular set intersection solution. Both the protocols use oblivious pseudorandom function evaluations instantiated as Diffie-Hellman based solutions, and the intersection list is obtained by comparing the OPRF evaluations on their input set elements. The security in both protocols is established using zero-knowledge proofs. They also claim that their solution for set intersection is much more efficient than [60], although both of them boast linear communication and computation complexities. A drawback of these solutions is that the exponentiations and multiplications involved in the protocols are performed on *1024*-bit numbers which naturally incurs a heavy computation cost.

Jarecki and Liu [61] bettered their earlier work [60], nevertheless with same communication and computation complexities, by replacing the OPRF with an Oblivious Unpredictable Function that can be executed parallelly for multiple set elements. Moreover, their proposed protocol for two-party set intersection, secure in the *malicious adversaries* model, drastically decreased the required

number of exponentiations, as compared to [60]. Also, since the exponentiations involve smaller, 160-bit exponents, the computational complexity of this protocol is expected to be at least 20 times less expensive than [60]. The security in the *malicious adversaries* model is established using zero-knowledge proofs, however, since only one party (*Receiver*) receives the final output, the other party (*Sender*) is the only one to use zero-knowledge proofs to prove his/her adherence to the protocol.

Ateniese et al. [7] improved the work by some of their authors in [30]. They introduced a protocol for two-party set intersection where the party obtaining the result can completely hide the size of his/her input set, including the upper bound. The approach, similar to [30], however has a non-linear computation complexity of $O(m \log m)$, where m is the size of each party's input set. While, they also suggested a modification to make the computation complexity linear, both their protocols are only secure under the *honest-but-curious adversaries* model. The privacy of the party not receiving the output is preserved due to the fact that only he/she knows the prime factors to the RSA modulo. Hence, restricting the other party from obtaining the OPRF evaluations for elements not in his/her list.

Cristofaro et al. [29] devised protocols solely the for cardinality of set intersection and union between two parties. These solutions follow a similar approach as [30], using oblivious pseudorandom function evaluations instantiated as Diffie-Hellman based solutions. In this work however, the link between the set elements and their evaluations are destroyed by randomly permuting the list of resulting evaluations. This way, only the cardinality of the resulting set can be established. Both the protocols suggested are to determine the set intersection cardinality, the union cardinality has to be computed using the result of the protocols, and the cardinalities of the individual set. One of the protocols requires one party to use authorized inputs as in [30]. Both the protocols enjoy linear communication and computation complexity, however they are only secure in the *honest-but-curious adversaries* model. Additionally, they show how their set intersection cardinality protocol in this work and their set intersection protocol from [30] can be combined. This way, one of the parties obtains the intersection cardinality of their sets, depending on which that party can choose to proceed to compute their set intersection or abort.

2.3 SECRET SHARING BASED SOLUTIONS

2.3.1 *Primitives*

SECRET SHARING Secret sharing are schemes through which a secret, let's say s, is shared between a number of parties. The only way to reconstruct the original value, s, is by collating the secret shares of preset combinations of the involved parties [99].

Shamir's Secret Sharing A t-out-of-n threshold scheme, Shamir's secret sharing splits and shares the secret value among n parties [98]. A combination of any t or more parties (with $t \leq n$) among them can reconstruct the secret. The shares are generated from polynomial evaluations in a finite field of prime order p. The secret generation phase requires the party with the secret, $s \in \mathbb{Z}_p$, to create a polynomial function, let's say f, of degree t-1 with $f(0) = s$ and the rest of the coefficients randomly chosen from $\mathbb{Z}_p$. Once the function is generated, the secrets are created of the form $(i, f(i))$ and distributed to parties $i = 1,\ldots,n$ respectively. The original secret can then be recomputed using the shares of at least t parties with the Lagrangian Interpolation formula,

$$s = f(0) = \sum_{i=1}^{t} f(x_i) \prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{x_j}{x_j - x_i}, \tag{9}$$

where x_i represents the first element of the share held by the i^{th} party, which is usually just i. Let the notation $\langle s \rangle_i$ imply the secret share of s held by the i^{th} party. Shamir's Secret Sharing scheme also has additively homomorphic properties, i.e. the i^{th} party can compute secret share of sum of two secrets values by summing the secret shares of the two secret values as $\langle s_1 + s_2 \rangle_i = \langle s_1 \rangle_i + \langle s_2 \rangle_i$. Moreover, the secret value can also be appropriately modified by adding or multiplying the secret shares with a public value, lets say k, i.e. $\langle s + k \rangle_i = \langle s \rangle_i + k$ and $\langle s \times k \rangle_i = \langle s \rangle_i \times k$.

2.3.2 *Existing Literature*

Li and Wu [68] devised the first information theoretically secure protocol for multi-party set intersection. The idea behind the protocol, similar to [67], is to compute the composite polynomial, representing the intersection of all n parties' sets, using every party's set representing polynomial. However, they resort to two-dimensional verifiable secret sharing [17, 55] (where shares of secret values are re-shared among them) instead of additively homomorphic encryption schemes. The protocol is secure in the presence of t passive adversaries or active adversaries, as long as $t < \frac{n}{2}$ or $t < \frac{n}{3}$ respectively. The work claims a communication complexity of $O(n^4k^2)$ (where k is the number of elements in each party's set)in the *malicious adversaries* model with active adversaries. While the communication complexity is very high, it is important to note that all the operations and communications are performed with secret values or shares.

Patra et al. [87] claimed that the communication complexity declared in [68] was $O(n^5k)$ rather than $O(n^4k^2)$ in the presence of a physical broadcast channel. They follow an approach similar to [68] with polynomial representation of sets and two-dimensional verifiable secret sharing [17, 55]. Their protocol for multi-party set intersection, under similar security assumptions as [68] in the active adversaries scenario, was $O(n^3k^2 + n^4)$. The same authors [88] also use a similar solution to construct a multi-party set intersection protocol information theoretically secure in the presence of t active

adversaries with an increased communication complexity of $O(n^4k^2 + n^5)$, as long as $t < \frac{n}{2}$ (a more relaxed constraint as compared to [87]). [68], [87] and [88], all rely on multiplications with randomly generated polynomials. However, the high communication complexity required for multiplication of secret shares detrimentally impacts the protocols' communication overhead.

Blanton and Aguiar [19] proposed a framework for computing a range of multi-party set operations, including set intersection, union and cardinality. Their protocols are based on secret sharing, and hence boast information theoretic security. The protocols suggested in the work are, in fact, simple algorithms built using efficient oblivious sorting [47], and multi-party logical operations using their secret shares [26]. They also propose optimizing the algorithms by pre-sorting them and performing merge sort [15] to combine them. Since their algorithms heavily depend on sorting the combined list before performing further operations, this vital optimization improves the performance of all the suggested protocols. The security in these protocols is established using verifiable secret sharing [5]. The protocols are secure in the presence of passive adversaries with a communication complexity $O(nlk \log k + n^2)$, when n parties with k elements each are involved and each element is represented using l bits, provided the adversaries control less than $\frac{n}{2}$ parties. Similarly, the protocols are secure in the presence of active adversaries with a communication complexity $O(nlk \log k + n^2 \log k + n^3)$, provided the adversaries control less than $\frac{n}{3}$ parties.

2.4 GARBLED CIRCUIT BASED SOLUTIONS

2.4.1 *Primitives*

A	B	$C = A \cap B$
0	0	0
0	1	0
1	0	0
1	1	1

(a) Logic Table

A	B	$C = A \cap B$
X_0^a	X_0^b	X_0^c
X_0^a	X_1^b	X_0^c
X_1^a	X_0^b	X_0^c
X_1^a	X_1^b	X_1^c

(b) Key Table

C
$X_0^a(X_0^b(X_0^c))$
$X_0^a(X_1^b(X_0^c))$
$X_1^a(X_0^b(X_0^c))$
$X_1^a(X_1^b(X_1^c))$

(c) Encrypted Table

C
$X_0^a(X_1^b(X_0^c))$
$X_1^a(X_1^b(X_1^c))$
$X_0^a(X_0^b(X_0^c))$
$X_1^a(X_0^b(X_0^c))$

(d) Garbled Table

Figure 4: Generating a Garbled Circuit to represent an *AND* gate [71]

GARBLED CIRCUITS Garbled Circuits are a family of cryptographic protocols that enable two mutually mistrusting parties to collectively evaluate a function with their private inputs, provided the function can be represented as a boolean circuit [71, 107]. Let the two involved parties be Alice and Bob, each with a secret bit y_a and y_b respectively, and the function to be evaluated be $y_a \cap y_b$. Alice first creates a boolean logic table for all different input values possible (as shown in Figure 4a) and replaces the bit values with appropriate randomly generated keys (as shown in Figure 4b). The keys' size is chosen based on the symmetric encryption scheme that is used in the following step. Alice then proceeds to encrypt the output keys with its corresponding input keys, sequentially, using a symmetric encryption scheme, such as AES [32], to create the encrypted table (as shown in Figure 4c). Then, Alice permutes the encrypted table and sends this "garbled" table (shown in Figure 4d) to Bob, along with her key $X^a_{y_a}$. Bob obtains his respective key $X^b_{y_b}$ through OT from Alice and finally decrypts all the outputs using the two obtained keys. Only one of the outputs can be decrypted correctly and that indicates the result.

2.4.2 *Existing Literature*

Huang et al. [57] disputed claims from previous work that, building solutions for secure set intersection using generic solutions such as Yao's Garbled Circuits [71, 107] would be inefficient. They built a number of protocols for two-party set intersection, secure in honest-but-curious adversary model, using Yao's Garbled Circuits [71, 107] framework. The protocols are, in fact, simple algorithms, in which only some of the components such as equality and greater/lesser than checking, and shuffling [106] are built using garbled circuits. Their most efficient protocol for large set sizes uses pre-sorted sets as inputs, obliviously merges them using sorting networks [15] and finally shuffles the list to prevent leaking information. Their most efficient solution computationally outperformed that of [31] in almost all considered security levels based on key sizes. The framework suggested in their protocols can also be used to compute more complex and arbitrary functions, which is advantageous.

2.5 BLOOM FILTER BASED SOLUTIONS

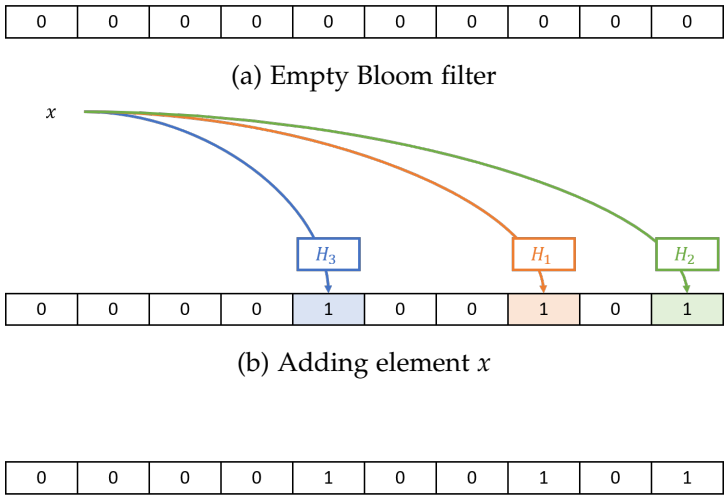
2.5.1 *Primitives*

CRYPTOGRAPHIC HASH FUNCTIONS A function, lets say H, which takes in bit strings of arbitrary length as input and produces fixed-length bit strings as outputs (often called digest, hashcode or hash value) is a Cryptographic Hash Function [99]. These functions are designed to be one-way functions, i.e. it should be infeasible to invert them. These functions should accommodate additional properties, which are explained below:

- **Deterministic -** The computation of the digest using the function has to be deterministic, i.e. the same input should always produce the same digest.
- **Pre-image resistant -** It should be computationally infeasible to compute the input from its digest.
- **Second pre-image resistant -** It should be computationally infeasible to identify two different inputs which produce the same digest.
- **Non-correlation -** There should be no correlation between the inputs and the digests, i.e. even a small change in the input should incur sufficient change in the digest such that no correlation is noticeable due to the change.

BLOOM FILTERS Bloom Filters are probabilistic and space-efficient data structures that are used to test if a set contains an element [20]. They are composed of a bit array, say BF, of preset length, say m, and a preset number of cryptographic hash functions, say $H_1, ..., H_\ell$. The two phases involved in utilizing Bloom filters are explained below:

- **Add an element -** In this phase, an element, say x, is added to the Bloom filter. The element is first fed to all the cryptographic hash functions. Subsequently, the positions in the bit array corresponding to $H_i(x) \mod m \ \forall i \in 1, ..., \ell$ are all set to 1. A visual example of this process is shown in Figure 5.



(a) Empty Bloom filter

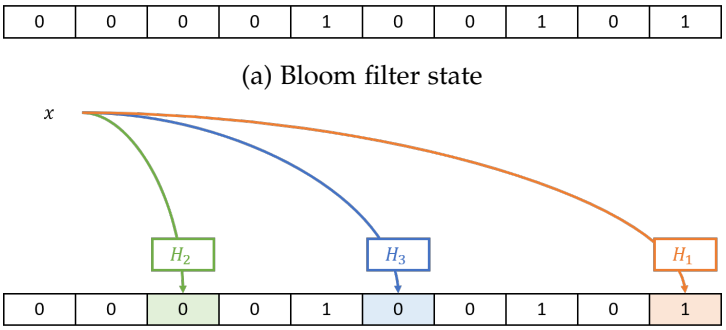
(b) Adding element x

(c) Bloom filter after addition

Figure 5: Adding an element x to a Bloom filter with $m = 10$, and $\ell = 3$

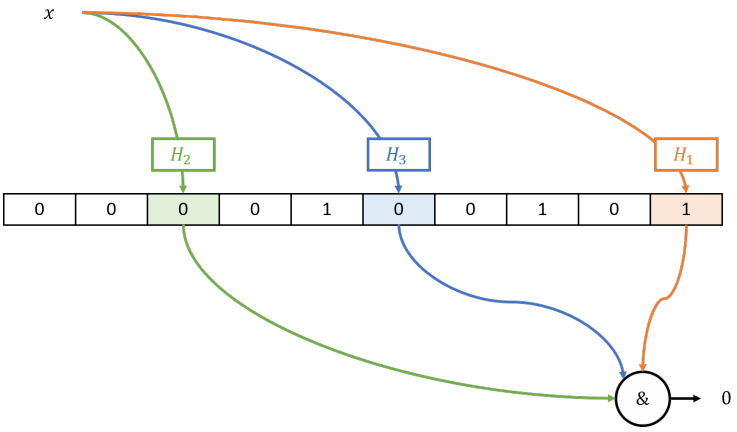
- **Query an element -** In this phase, the membership of an element, say x, in a Bloom filter is tested. Similar to adding, the element is first fed to all the cryptographic hash functions. With the digests produced by

functions, the membership of the element is computed using the formula $\wedge_{i=1}^{\ell} B(H_i(x) \mod m))$. The result of which is 0 if the element is not present and 1 if it is. Where *B(k)* is the value of the k^{th} bit in the bit array B.



(a) Bloom filter state

(b) Querying element x



(c) Obtaining result

Figure 6: Querying an element x in a Bloom filter with $m = 10$, and $\ell = 3$

The probabilistic nature of Bloom Filters is attributed to the fact that the membership test for elements is not deterministic. There is a likelihood that the membership test could produce wrong results, but this is contained to the cases when the membership test is positive. If the membership test for an element is negative, then the element is absolutely not present in the filter. The false positive rate ϵ is dependent on the number of elements added to the filter, k^*, the length of the bit array, m, and the number of cryptographic hash functions used, ℓ. This dependence can be mathematically represented as,

$$\epsilon = (1 - (1 - \frac{1}{m})^{n\ell})^{\ell} \approx (1 - e^{-n\ell/m})^{\ell}. \tag{10}$$

The equations for optimal m and ℓ are given in Equation 11 and Equation 12, respectively [37]. In this paper we always assume the optimal values for m and ℓ, unless stated otherwise.

$$m = -\frac{k^* \ln \epsilon}{(\ln 2)^2} \tag{11}$$

$$\ell = \frac{m}{k^*} \ln 2 \tag{12}$$

2.5.2 *Existing Literature*

Dong et al. [37] used a novel approach called Oblivious Bloom Intersection to devise protocols for two-party set intersection. Their protocols are based on obtaining the Garbled Bloom Filter (a variant of Bloom filter proposed by them) representing the intersection of both their sets. The solution, secure in *honest-but-curious adversaries* model, comprised of simply evaluating the required garbled Bloom filter with a number of parallel OTs, using one party's Bloom filter, and the other party's garbled Bloom filter as input. This protocol is modified to be secure in *malicious adversaries* model using symmetric key encryption, XOR-based secret sharing, and OTs secure in *malicious adversaries* model. The implementation of their solution outperformed the protocols suggested in [31] and [37] by orders of magnitude in the two security levels (based on key size) considered. It is, however, important to note that there is a likelihood that the process of initializing the garbled Bloom filter could result in a failure due to inability to accommodate a new set element. Also, in their protocol for the *malicious adversaries* model, there are possibilities to deviate from the protocol, such as by not correctly computing the secret shares.

Pinkas et al. [91] improved the efficiency of the work proposed by [37], in *honest-but-curious adversaries* model, by modifying the protocol to random garbled Bloom filter evaluations. Instead of constructing a garbled Bloom filter as suggested by [37], they proposed that one of the parties initializes a garbled Bloom filter like structure (of array size m) with random values $r_1, ..., r_m$. The other party follows the protocol as in [37], but instead obtains r_i or 1 depending on whether the ith position in his/her Bloom filter is 1 or 0, respectively. Subsequently, the random values in positions corresponding to an element are XORed to obtain the oblivious garbled Bloom filter evaluation for that element. The intersection is found by comparing the evaluations of both parties' sets. A huge drawback of this method is that the party who creates the random garbled Bloom is the one to make the comparison between the evaluations, which implies that that party can view all the evaluations of the other party. These evaluations can then be tied to their original set elements by comparing them with evaluations of all possible elements using brute-force.

2.6 OBLIVIOUS TRANSFER BASED SOLUTIONS

2.6.1 *Existing Literature*

Pinkas et al. [91] suggested a protocol for two-party set intersection using balanced hashing [10] and a variant of OT called random OT [6]. Similar to their suggested modification to [37], this protocol is based on comparing the evaluations of the random OT protocol on their set elements. This protocol is only secure in the *honest-but-curious adversaries* model, as is the other protocol suggested by them. However, comparing their implementation with [37], has shown theirs to be considerably faster.

2.7 OTHER SOLUTIONS

2.7.1 *Primitives*

TRUSTED EXECUTION ENVIRONMENT Trusted Execution Environment (TEE) is a secure segregation at a processor and memory level in a system [103]. It protects the security-critical logic from operations originating outside the TEE. All software and memory outside the TEE belong to the Rich Execution Environment (REE), which also contains the operating system and the bulk of other software in the system. Applications running in the TEE and REE are called Trusted Applications (TA) and Client Applications (CA), respectively. The concept of a TEE is to isolate processes and information stored for secure functions, so that even when the system is compromised, no software outside the TEE can interfere with TA's operation. The idea is motivated by attacks on the hardware of a system using its software, such as in [97].

2.7.2 *Existing Literature*

Tamarkar et al. [103] suggested using trusted hardware for two-party set intersection problem. They proposed using Trusted Execution environments to perform security-critical operations. In their system setting, one of the parties is a server which contains the TEE. A dictionary of the items in the server's set is maintained in its REE. The protocols proposed in this work aim to access the dictionary items in an efficient manner without revealing the other party's (Client) input elements. The protocols structure the dictionary differently, viz. Sequence of differences, Bloom filter and 4-ary Cuckoo Hash [38, 85], to process it efficiently and privately. In addition to that, they improve the security of the protocols by also implementing them along with ORAM Schemes [46], which divides and encrypts data into blocks, and stores them in a randomized manner. The protocols were implemented, under similar initial constraints, in the two most commercially prevalent TEEs, viz. Intel SGX and ARM TrustZone. The protocol utilizing 4-ary Cuckoo hashing without ORAM proved to be the most efficient in both the platforms. An important advantage of these protocols is that the lack of cryptographic preliminaries allows them to be efficient.

2.8 COMPARATIVE ANALYSIS AND DISCUSSION

A comparative analysis of the communication and computation complexities of the discussed solutions is presented in Table 1. Linear communication and computation complexities for two-party settings have been achieved by a number of works, even in the *malicious adversaries* model [30, 60, 61]. However, only a small portion of the works in this field have considered the multi-party setting. Among these, the solutions proposed by Blanton and Aguiar [19] have the least communication and computation complexity. Moreover, their solutions are based on secret sharing, and hence command unconditional security. Their work also involves less expensive computations and allows for construction of generic solutions.

Interestingly, the newer works in this field [37, 57, 91] compare the performances of their implementation, rather than their asymptotic complexities, with that of the most efficient previous work. This choice is understandable given the fact that these works use generic techniques such as Yao's Garbled Circuits [57], OTs [91] and Bloom filters [37], which reduces the requirement of computationally heavy operations, such as exponentiations. Also these solutions, in spite of having higher asymptotic complexity, are often faster than the earlier ones, if constructed appropriately [57]. Kiss and Liu [66] implemented and compared two-party set intersection protocols, secure in HBC model, based on RSA [31], Diffie-Hellman [58], Naor-Reingold OPRF [51, 82] and AES [32] based Garbled Circuits [92]. All the mentioned protocols, except for the Garbled Circuits based one, have linear communication and computation complexities. Under the same initial constraints, the solution based on garbled circuits has the best overall runtime. This validates the claim in [57] that, efficient solutions are plausible using generic solutions such as Garbled Circuits [71, 107]. However, it is important to note that among the new protocols only [37] has solutions that are secure in the *malicious adversaries* model and even that has flaws in ensuring the correct implementation of the protocol. Security, for the protocols secure in *malicious adversaries* model, is established predominantly using multiple zero-knowledge proofs, or cut-and-choose. Both these techniques incur high communication and computation overheads. Hence, it is important to strike a balance between efficiency and security of the protocol. The adversary models assumed in [51] provides a plausible solution to approach this problem.

2.9 OPEN ISSUES

As stated previously, the set union-combinatorial intersection cardinalities can be evaluated using privacy-preserving protocols to compute its two constituent set operations, viz. multi-party set union and two-party set intersection cardinality. However doing so, for n involved parties, would require $2^{n-1} - n$ iterations of multi-party set union operation and $2^{n-1} - 1$ iterations of two-party set intersection cardinality operation. Hence, necessitating protocols to compute composite set operations, which require evaluating the result of a

Category	*Authors*	*Adv. Model*	*Asymptotic Complexity*	
			Computation	*Communication*
HE	[41]	HBC	$\mathcal{O}(k)$	$\mathcal{O}(k \ln \ln k)$
	[67]	HBC	$\mathcal{O}(cnk^2)$	$\mathcal{O}(cnk)$
		Mal	$\mathcal{O}(n^2k^3)$	$\mathcal{O}(n^2k)$
	[42]	HBC	$\mathcal{O}(n^2k^2)$	$\mathcal{O}(n^2k)$
		Mal	$\mathcal{O}(n^2k^3)$	$\mathcal{O}(n^2k^2 + n^3k)$
	[53]	Mal	$\mathcal{O}(k(\log \log k + \ell))$	$\mathcal{O}(k\ell)$
OPRF	[51]	OSS	$\mathcal{O}(k\ell)$	$\mathcal{O}(k\ell)$
		Cov	$\mathcal{O}(k\ell)$	$\mathcal{O}(k\ell)$
	[60]	HBC	$\mathcal{O}(k)$	$\mathcal{O}(k)$
		Mal	$\mathcal{O}(k)$	$\mathcal{O}(k)$
	[30]	Mal	$\mathcal{O}(k)$	$\mathcal{O}(k)$
	[61]	Mal	$\mathcal{O}(k)$	$\mathcal{O}(k)$
	[7]	HBC	$\mathcal{O}(k)$	$\mathcal{O}(k)$
	[29]	HBC	$\mathcal{O}(k)$	$\mathcal{O}(k)$
SS	[68]	Mal	$\mathcal{O}(n^4k^2)$ **or** $\mathcal{O}(n^5k)$ †	$\mathcal{O}(n^4k^2)$ **or** $\mathcal{O}(n^5k)$†
	[87]	Mal	$\mathcal{O}(n^3k^2 + n^4)$	$\mathcal{O}(n^3k^2 + n^4)$
	[88]	Mal	$\mathcal{O}(n^4k^2 + n^5)$	$\mathcal{O}(n^4k^2 + n^5)$
	[19]	Mal	$\mathcal{O}(nk \log k + n^2 \log k + n^3)$	$\mathcal{O}(nk \log k + n^2 \log k + n^3)$
GC	[57]	HBC	$\mathcal{O}(k \log k)$	$\mathcal{O}(k \log k + \lambda)$
BF	[37]	HBC	$\mathcal{O}(-k\lambda \ln \epsilon)$	$\mathcal{O}(-k \ln \epsilon + k\lambda)$
		Mal	$\mathcal{O}(-k\lambda \ln \epsilon)$	$\mathcal{O}(-k \ln \epsilon + k\lambda)$
	[91]	HBC	$\mathcal{O}(-k\lambda \ln \epsilon)$	$\mathcal{O}(k(\log k + \lambda))$
OT	[91]	HBC	$\mathcal{O}(k \log k)$	$\mathcal{O}(k\ell + k^2 \log k)$

Table 1: Comparative analysis of prior art (n: No. of participants; k: No. of items in each participant's set; c: Maximum no. of dishonestly colluding parties; ℓ: No. of bits used to represent each set element; λ: Security Parameter; ϵ: Bloom filter false positive rate)

† *Complexity estimation by [68]* **or** *Complexity estimation by [87]*

set operation between a particular set and all combinations of a number of other sets; which we term as a *combinatorial set operation*. A combinatorial set operation is composed of two set operations: the first operation to compute a single set resulting from all sets in a combination, and second to evaluate the result between the particular set and all different combinations. *Set union-combinatorial intersection cardinality* is one such combinatorial set operation. The union operation in the terminology is used to obtain a single

set from all sets in a combination. The *intersection cardinality* is used to evaluate the result between the particular set and each combination. Our research question can be addressed by constructing a protocol to privately compute set union-combinatorial intersection cardinality between the *client's* set and the *vendors'* sets.

The most efficient solution for two-party private set intersection cardinality is by De Cristofaro et al. [29], which has computation and communication complexities linear to the cardinality of the input sets. The protocol is secure in the *honest-but-curious adversaries* model, and the result of the protocol is only revealed to one party, which is appropriate for our problem. The most efficient solution for multi-party private set union is by Blanton and Aguiar [19] whose protocols have information-theoretic security. Their solutions endure $\mathcal{O}(nbk^* \log(k^*) + n^2)$ and $\mathcal{O}(nbk^* \log(k^*) + n^2 \log(k^*) + n^3)$ asymptotic complexities in the presence of passive and active adversaries respectively. The asymptotic complexities are computed for n parties, where k^* is the combined cardinality of the input sets and b is the bit length of a set element.

Clearly, even the most efficient protocols are not sufficient to design a feasible solution to compute set union-combinatorial intersection cardinality since the resulting protocol's complexity would be exponentially dependent on the number of parties. Moreover, the protocol would compromise on the privacy of the involved parties due to the presence of intermediate transcripts, such as the results of the multi-party set union for each combination. Hence, we aim to devise protocols to compute the set union-combinatorial intersection cardinality without leaking any information about the input set elements, and whose asymptotic complexities have a quadratic or lower dependence on the number of parties.

3

PRELIMINARY PROTOCOLS

In this chapter, we introduce and analyse three subprotocols used in our principal protocol which are

1. Private Set Union-Combinatorial Intersection Cardinality in *honest-but-curious adversaries* model,
2. Secure Simple Multi-party Coin Tossing in *malicious adversaries* model, and
3. Secure Shuffle and Exponentiate in *covert adversaries* model.

The notations used in this thesis is explained in Table 2.

Symbol	*Explanation*
$\mathcal{P}$	The set of participants, s.t. $\mathcal{P} = \{P_1, \ldots, P_n\}$
P_1	The client
$\mathcal{V}$	The set of vendors, s.t. $\mathcal{V} = \{P_2, \ldots, P_n\}$
S_i	The set held by P_i
k_i	Cardinality of S_i s.t. $\lvert S_i \rvert = k_i$
K_i	P_i's secret exponentiation key
Q_i	P_i's final exponentiated set
K_i^L	P_i's *left* half-key
K_i^R	P_i's *right* half-key
BF_i	P_i's Bloom filter array
Y_i	P_i's random bit-string
r_i	P_i's random number
$S_i[j]$	jth item in S_i
S_i^u	P_i's set after u rounds of exponentiation
N	Prime modulus of all operations
$\phi(N)$	Totient function of N
h, g	Generators in the prime order N
$PC_r(m)$	Pedersen's Commitment on m (= $h^m g^r$)

Table 2: Notation scheme

All the three preliminary protocols share the similar initial settings. The protocol comprises of n parties, $(P_1, ..., P_n)$, of whom P_1 is the *client* and the rest are *vendors*. Each party P_i has a set S_i with k_i items. All involved parties are privy to the prime modulo of operations N and are connected together in a common *broadcast channel*. Additionally, all parties are also aware of the

common public parameters of Pedersen commitment scheme (g, h, N), where g and h are generators of cyclic groups with prime modulo N. Additionally, in private set union-combinatorial intersection cardinality in *honest-but-curious adversaries* Model, the parties are organized in a circular structure with each party P_i, $i \in [1, n]$ is preceded by P_{i-1} and succeeded by P_{i+1}, where P_0 corresponds to P_n, and P_{n+1} corresponds to P_1. The circular structure is shown in Figure 7.

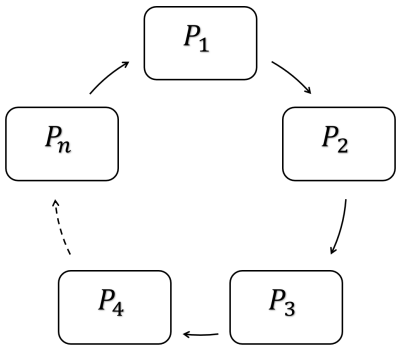


Figure 7: The circular structure showing the sequence of computation among the parties

3.1 PRIVATE SET UNION-COMBINATORIAL INTERSECTION CARDINALITY IN HONEST-BUT-CURIOUS ADVERSARIES MODEL (SUCCINCT-HBC)

SUCCINCT-HBC aims to obtain the intersection cardinality between *client's* set and all union combination of sets held by a finite number of *vendors'*, in *honest-but-curious adversaries* model. The result of the protocol is only exposed to the *client*. Protocol 4 describes the specifications of SUCCINCT-HBC. The protocol is split into five phases, viz. *Offline Phase*, *Posting Phase*, *Exponentiation Phase*, *Transmission Phase* and *Combinatorial Phase*. In the *Offline Phase*, all parties choose their respective secret keys and *half*-keys. Subsequently, in the *Posting Phase* all parties post their set elements exponentiated with either of their *half*-keys. All parties then cyclically exponentiate the set elements of other parties with their secret keys in the *Exponentiation Phase*. After which, in the *Transmission Phase*, each party completes exponentiating their set elements with their other *half*-key. After which, all parties except the *client* individually create and transmit the Bloom filter representation of their exponentiated set to the *client* in a secure channel. Finally, in the *Combinatorial Phase*, the *client* computes the membership matrix $\mathcal{M}$, which is the result of SUCCINCT-HBC. For a union combination $\delta \subset \mathcal{V}$, whose union set is represented by S_δ, $|S_1 \cap S_\delta|$ can be computed using the formula,

$$|S_1 \cap S_\delta| = \sum_{j=1}^{k_1} (\cup_{P_i \in \delta} (\mathcal{M}_i[j])). \tag{13}$$

The union-combinatorial intersection cardinality, between the *client's* set and the *vendors'* sets, can be extracted from $\mathcal{M}$. However, since there are $2^{n-1} - n$

combinations, any measure to evaluate the required union-combinatorial intersection cardinality from $\mathcal{M}$ would have computation complexity of at least $\mathcal{O}(2^n)$. Thus special optimizations have to be employed to efficiently obtain the required combination of *vendors* from $\mathcal{M}$. An optimization algorithm to obtain a smallest subset of *vendors* whose combined union set has maximum intersection cardinality with the *client's* set is presented in Chapter 6.

The intuition behind SUCCINCT-HBC is to enable all parties to jointly evaluate $\mathcal{F}_K(s) = s^K$ for each element s in their set, where $K\,(= \prod_{i=1}^{n} K_i)$ is the joint secret key between all the parties. No party should be able to compute K without all parties working together. $\mathcal{F}$ is chosen to be an exponentiation function due to its commutative property, similar to the hashing employed in [105]. This function can be replaced with any other commutative operation, provided it is information-theoretically or computationally infeasible to retrieve the secret key from the input and output transcripts of the operation.

Protocol 4 : SUCCINCT-HBC

1. **Offline Phase -** Each party $P_i,\ i \in [1, n]$
   a) Chooses a secret key $K_i \in \mathbb{Z}^*_{\phi(N)}$.
   b) Picks $K_i^L \in_R \mathbb{Z}^*_{\phi(N)}$ and computes $K_i^R = K_i \times {K_i^L}^{-1} \mod \phi(N)$ such that $K_i^L \times K_i^R = K_i \mod \phi(N)$.
2. **Posting Phase -** In a broadcast channel, each party $P_i,\ i \in [1, n]$ computes and posts $S_i^1 = \{(S_i[j])^{K_i^L} |\ j \in [1, k_i]\}$.
3. **Exponentiation Phase -** All parties engage in cyclic exponentiation of the posted elements for *n-1* rounds. In round $u,\ u \in [1, n-1]$, each set $S_i^u\ i \in [1, n]$ is exponentiated by the party P_{i+u}, who creates the new list $S_i^{u+1} = \Omega(\{(S_i^u[j])^{K_{i+u}} |\ j \in [1, k_i]\})$, where $\Omega()$ is a random permutation function.
4. **Transmission Phase -** Each party $P_i,\ i \in [1, n]$
   a) exponentiates the set S_i^n with her key K_i^R to obtain the set $Q_i = \{(S_i^n[j])^{K_i^R} |\ j \in [1, k_i]\}$.
   b) creates the Bloom filter of her set, BF_i, with k^* as the overall cardinality and a mutually accepted ϵ as the false positive rate, where $k^* = \sum_i k_i,\ i \in [1, n]$. Then, transmits BF_i to P_1 in a secure channel. P_1 does not perform this action.
5. **Combinatorial Phase -** P_1 computes a membership matrix $\mathcal{M}$, s.t. $\mathcal{M}_i[j]$ represents the result of the membership test of element $Q_1[j]$ in the Bloom filter digest BF_i, $\forall\, i \in [2, n]$. The result of $\mathcal{M}_i[j]$ is *1* if the membership test is positive and *0* if negative.

Our choice to use *half*-keys is motivated by the fact that if all parties exponentiated their set elements with their full keys instead, at the end of the *Exponentiation Phase* every set element of all parties would be exponentiated with the same value. Since the exponentiated items are visible to all parties, any party can obtain the intersection and/or union cardinalities between any two parties or group of parties. We also choose to transmit the final set representations of *vendors* using Bloom filters instead of the exponentiated sets, despite the fact that it increases the communication complexity. In both cases, the *client* can losslessly compute all union combinations of all exponentiated sets. Bloom filter representations ensure that the *client* only learns about the membership of her own elements in the union combinations and not about the relation between the individual sets. By using *half*-keys and Bloom filters, we ensure that the *client* or the *vendors* cannot learn about the relation between other parties' sets. One drawback, however, is that at the end of the protocol, the *client* can make approximate estimations of the intersection cardinality of any two sets given their Bloom filter digests [102]. It does not, however, leak any information about the set elements themselves. Additionally, we assume that there is no collusion between the *client* and any *vendor* since a *vendor* can compute the union-combinatorial intersection cardinality between her set and the other *vendors*' in case of a collusion. Both these issues have also been resolved in Section 5.2.

3.1.1 *Asymptotic complexity analyses*

To compute the computational and communicational complexities of our protocols, we assume that all n parties involved have k elements in their sets. However, the protocols are not restricted by any such requirements and can endure each involved party having a different set cardinality. Additionally, let the false positive rate be ϵ, the size of the final Bloom Filter be m (where $m = -\frac{k^* \ln \epsilon}{(\ln 2)^2} = -\frac{nk \ln(\epsilon)}{(\ln 2)^2}$) and the number of hash functions for the Bloom filter be ℓ (where $\ell = \frac{m}{k^*} \ln 2 = \frac{\ln \epsilon}{\ln 2}$). The computational complexities estimate only the number of exponentiation and hashing operations as the other operations require negligible processing time in comparison.
Table 3 provides the computation and communication analyses of the first protocol. The most expensive round of the protocol, for both computation and communication, is *Step 5*, as it involves *n-1* rounds. The individual computation and communication complexities for each party is the same except for the communication in *Step 5*, where the *client* does not have to transmit any message. The computation complexity is $\mathcal{O}(k(n+\ell))$ or $\mathcal{O}(k(n-\ln(\epsilon)))$ per party and $\mathcal{O}(n^2k + nk\ell)$ or $\mathcal{O}(n^2k - nk\ln(\epsilon))$ overall, and the communication complexity is $\mathcal{O}(nk+m)$ or $\mathcal{O}(nk(1-\ln(\epsilon)))$ per party and $\mathcal{O}(n(nk+m))$ or $\mathcal{O}(n^2k(1-\ln(\epsilon)))$ overall.

		Computation Complexity		*Communication Complexity*	
		Each Party	*Total*	*Each Party*	*Total*
Step 1	*a*	-	-	-	-
	b	1	n	-	-
Step 2		k	nk	k	nk
Step 3		$(n\text{-}1)k$	$n(n\text{-}1)k$	$(n\text{-}1)k$	$n(n\text{-}1)k$
Step 4	*a*	k	nk	-	-
	b	- ***or*** $k\ell^{\dagger}$	$k\ell(n\text{-}1)$	- ***or*** $m^{\dagger}$	$(n\text{-}1)m$
Step 5		$k\ell$ ***or*** -†	$k\ell$	-	-
Overall		$(n+\ell+1)k + 1$	$n(n+1)k+nk\ell+n$	nk ***or*** $nk+m^{\dagger}$	$n^2k + (n-1)m$

Table 3: Asymptotic complexity analysis of SUCCINCT-HBC

† *Estimation for Client* **or** *Estimation for Vendors*

3.1.2 *Security analysis*

To prove the security of SUCCINCT-HBC, we show that the view of a non-adaptive, computationally-bound adversary $\mathcal{A}$ in the real model is computationally indistinguishable from the view of a probabilistic polynomial time simulator in the ideal model. Let us first define the execution of SUCCINCT-HBC in the ideal and real models:

EXECUTION IN THE IDEAL MODEL - All participants send their inputs to a trusted third party, who computes and dissipates the output to all relevant parties, in this case just the *client* (P_1). While the output of the protocol is considered to be $\mathcal{M}$, for the security analysis we will consider the output to be the Bloom filter digests of all the *vendors*. This consideration is due to the fact that $\mathcal{M}$ is just the result refined from the *vendors'* Bloom filters and the *client's* final exponentiated set Q_1. All information necessary to the output is, thus, contained in the Bloom filters and Q_1 It is more pertinent to treat them as the result of the protocol for the security analysis as they are more likely to leak information than the refined result $\mathcal{M}$. Let the functionality f^{π_1} compute the result of SUCCINCT-HBC in the ideal model, using the input sets $S_1, \ldots, S_n$ held by parties $P_1, \ldots, P_n$ and $f_i^{\pi_1}$ represent the output of the partial party P_i. Then,

$$f^{\pi_1}(S_1, \ldots, S_n) \text{ or } f^{\pi_1}(\bar{S}) = \{f_1^{\pi_1}(\bar{S}), \ldots, f_n^{\pi_1}(\bar{S})\} \tag{14}$$

and since only the *client* receives the result of SUCCINCT-HBC,

$$\textbf{output}^{\pi_1}(\bar{S}) = f^{\pi_1}(\bar{S}) = f_1^{\pi_1}(\bar{S}) = \{BF_i \mid i \in [2, n]\}. \tag{15}$$

EXECUTION IN THE REAL MODEL - In the real model, there is no trusted third party. Honest parties follow the protocol specification, and an adversary $\mathcal{A}$ interacts with the rest, in the place of corrupted party (or parties). The output of the real model is also considered to be the Bloom filters and Q_1 for the sake of the security analysis.

Let Sim_1 and Sim_2 be two probabilistic polynomial time simulators, we prove the security for $\mathcal{A}$ controlling a group of *vendors* and the *client*, separately.

GROUP OF VENDORS ARE CORRUPTED Let $\mathcal{A}$ control a group of *vendors*, $I \subset \mathcal{V}$, and $\bar{S}_I$, $f_I^{\pi_1}(\bar{S})$ and $\mathbf{view}_I^{\pi_1}(\bar{S})$ represents the set of inputs, outputs and views of all the *vendors* in I, respectively. We prove that

$$\{Sim_1(I, \bar{S}_I, f_I^{\pi_1}(\bar{S})),\ f^{\pi_1}(\bar{S})\} \stackrel{c}{\equiv} \{\mathbf{view}_I^{\pi_1}(\bar{S}),\ \mathbf{output}^{\pi_1}(\bar{S})\}. \tag{16}$$

Since all parties involved in the protocol can see all the intermediate transcripts, the view for each party P_i can be defined as

$$\mathbf{view}_i^{\pi_1}(\bar{S}) = (K_i, K_i^L, K_i^R, S_i, S_1^q, \ldots, S_n^q),\ \forall q \in [1, n],\ and \tag{17}$$

$$\mathbf{view}_I^{\pi_1}(\bar{S}) = \{\mathbf{view}_i^{\pi_1}(\bar{S}) \mid P_i \in I\}. \tag{18}$$

The simulator Sim_1 has all the input and output of all the parties in I. The simulator generates $\mathcal{K}_i$, $\mathcal{K}_i^L$, $\mathcal{K}_i^R$, $\mathcal{S}_1^q, \ldots, \mathcal{S}_n^q \in_R \mathbb{Z}_N$, $\forall q \in [1, n]$. Let $\Psi_I = \{\Psi_i \mid P_i \in I\}$, $\Psi \in \{\mathcal{K}, \mathcal{K}^L, \mathcal{K}^R\}$. The view simulated by Sim_1 is,

$$Sim_1(I, \bar{S}_I, \mathcal{F}_I^{\pi_1}(\bar{S})) = (\mathcal{K}_I, \mathcal{K}_I^L, \mathcal{K}_I^R, \mathcal{S}_I, \mathcal{S}_1^1, \ldots, \mathcal{S}_n^1, \ldots, \mathcal{S}_1^n, \ldots, \mathcal{S}_n^n). \tag{19}$$

The output observed by both the simulator Sim_1 and the adversary $\mathcal{A}$ is

$$\mathbf{output}_I^{\pi_1}(\bar{S}) = f_I^{\pi_1}(\bar{S}) = \perp, \tag{20}$$

as only the *client* receives the output in SUCCINCT-HBC. Then

$$\{Sim_1(I, \bar{S}_I, f_I^{\pi_1}(\bar{S})),\ f^{\pi_1}(\bar{S})\} \stackrel{c}{\equiv} \{\mathbf{view}_I^{\pi_1}(\bar{S}),\ \mathbf{output}^{\pi_1}(\bar{S})\}, \tag{21}$$

if for every non-uniform polynomial time distinguisher *Dis* there exists a negligible function $\mu(\kappa)$ such that

$$\begin{aligned} |[Pr(Dis((\mathcal{K}_I, \mathcal{K}_I^L, \mathcal{K}_I^R, \mathcal{S}_I, \mathcal{S}_1^1, \ldots, \mathcal{S}_n^1, \ldots, \mathcal{S}_1^n, \ldots, \mathcal{S}_n^n) \cap (\perp)) = 1] - \\ [Pr(Dis(\mathbf{view}_I^{\pi_1}(\bar{S}) \cap (\perp))) = 1]| \leq \mu(\kappa). \end{aligned} \tag{22}$$

Equation 22 holds under the discrete logarithm assumption [99] for any subset $I\ (\subset V)$, with κ being the security parameter intrinsic to the bit-sizes of N, K_i, K_i^L, and K_i^R. Hence, SUCCINCT-HBC is secure in the *honest-but-curious adversaries* model against any group of corrupted *vendors*.

CLIENT CORRUPTED Following the notation from the previous scenario, let $\mathcal{A}$ control the *client* (P_1), the view of the *client* be

$$\mathbf{view}_1^{\pi_1}(\bar{S}) = (K_1, K_1^L, K_1^R, S_1, S_1^q, \ldots, S_n^q),\ \forall\, q \in [1, n]. \tag{23}$$

Similarly, let the view simulated by the probabilistic polynomial time Sim_2 be

$$Sim_2(S_1, f_1^{\pi_1}(\bar{S})) = (\mathcal{K}_1, \mathcal{K}_1^L, \mathcal{K}_1^R, \mathcal{S}_1, \mathcal{S}_1^1, \ldots, \mathcal{S}_n^1, \ldots, \mathcal{S}_1^n, \ldots, \mathcal{S}_n^n), \tag{24}$$

where $\mathcal{K}_1$, $\mathcal{K}_1^L$, $\mathcal{K}_1^R$, $\mathcal{S}_1^q, \ldots, \mathcal{S}_n^q \in_R \mathbb{Z}_N$, $\forall\, q \in [1, n]$ are generated by Sim_2. Then,

$$\{Sim_2(\bar{S}_1, f_1^{\pi_1}(\bar{S})),\ f^{\pi_1}(\bar{S})\} \stackrel{c}{\equiv} \{\mathbf{view}_1^{\pi_1}(\bar{S}),\ \mathbf{output}^{\pi_1}(\bar{S})\}, \tag{25}$$

if for every non-uniform polynomial time distinguisher *Dis* there exists a negligible function $\mu(\kappa)$, such that

$$\begin{aligned} |[Pr(Dis((\mathcal{K}_1, \mathcal{K}_1^L, \mathcal{K}_1^R, \mathcal{S}_1, \mathcal{S}_1^1, \ldots, \mathcal{S}_n^1, \ldots, \mathcal{S}_1^n, \ldots, \mathcal{S}_n^n) \cap \{BF_2, \ldots, BF_n\})) = 1] \\ -[Pr(Dis(\mathbf{view}_I^{\pi_1}(\bar{S}) \cap \{BF_2, \ldots, BF_n\})) = 1]| \leq \mu(\kappa). \end{aligned} \tag{26}$$

Equation 26 holds under the discrete logarithm assumption [99], with κ being the security parameter intrinsic to the bit-sizes of N, K_i, K_i^L, and K_i^R. Furthermore, since the *client* never discovers the cumulative secret key K, $\mathcal{A}$ cannot brute-force for the set elements in the Bloom filter representation of the *vendors'* sets. Thus, SUCCINCT-HBC is secure in the *honest-but-curious adversaries* model against a corrupted *client*.

3.2 SECURE SIMPLE MULTI-PARTY COIN TOSSING IN MALICIOUS ADVERSARIES MODEL (SCULPT)

The first preliminary protocol has to be made secure in the *covert adversaries* model to obtain our principal protocol. It requires us to augment additional

Protocol 5 : SCULPT

Additional Setup: All parties are informed of the bit-string length, t.

1. **Offline Phase -** Each party P_i, $i \in [1, n]$ choses a random bit-string of length t, $Y_i \in_R \{0, 1\}^t$.
2. **Commitment Phase -** Each party P_i, $i \in [1, n]$ creates a Pedersen commitment, $PC_{v_i}(Y_i) = h^{Y_i} g^{v_i}$, after picking $v_i \in_R \mathbb{Z}_N^*$, and publishes it in the broadcast channel.
3. **De-commitment Phase -** Once all the parties have published their commitments, all parties decommit by sharing the parameters used to create their commitments (Y_i, v_i).
4. **Bit-string Computation Phase -** Finally, once the commitments are verified, all parties individually compute $R = Y_i \oplus \ldots \oplus Y_n$.

protocols to the first protocol, one of which is *Secure Simple Multi-party Coin Tossing*. SCULPT aims to compute a random bit-string of a fixed length,t, jointly by multiple mistrusting parties. The protocol, modified from the two-party basic coin-tossing protocol in [70], is secure in the *malicious adversaries* model, with all parties jointly evaluating the result. Protocol 5 describes how SCULPT is executed. The protocol consists of four phases, viz. *Offline Phase*, *Commitment Phase*, *De-commitment Phase* and *Bit-string Computation Phase*. In the *Offline Phase* all parties choose a random bit-string of length t. All parties then compute and post a Pedersen Commitment on their bit-strings in the *Commitment Phase*. Once all the parties have posted their commitments, they reveal the secret values used to generate the commitment by posting them in the *broadcast channel* during the *De-commitment Phase*. Finally, all parties individually compute the result in the *Bit-string Computation Phase* by XORing the bit-strings posted by all parties.

The intuition behind the protocol is that by committing to their bit-strings before revealing it, all parties are restricted to their chosen bit-strings. In such a scenario, the resulting bit-string is random even if only one party honestly generates a random bit-string. Additionally, we choose the *XOR* operation to obtain the result, as *XOR*ing multiple privately-generated, unknown bits could result in a 0 or 1 with equal likelihood. Thus, using such an unbiased operation ensures that the resulting bit-string is random.

3.2.1 *Asymptotic complexity analyses*

Table 4 shows the step-wise computation and communication analyses of the protocol. Both the asymptotic complexities are $\mathcal{O}(n)$, i.e. linear with respect to the number of parties involved. Moreover, the length of resulting bit-string has no influence on the complexities.

	Computation Complexity		***Communication Complexity***	
	Each Party	*Total*	*Each Party*	*Total*
Step 1	-	-	-	-
Step 2	2	$2n$	1	n
Step 3	-	-	2	$2n$
Step 4	-	-	-	-
Overall	2	$2n$	3	$3n$

Table 4: Asymptotic complexity analyses of SCULPT

3.2.2 *Security analysis*

Since there are no private elements in SCULPT, proving the correctness of the protocol is sufficient. Correctness requires that no party or a group of parties is able to manipulate the protocol in a way that they can craft their desired result. We should ensure that no party has access to the secret bit-strings of all other parties before choosing her own. Thus, all parties post a Pedersen Commitment

on their bit-strings prior to revealing their bit-strings. Learning about the other parties' bit-strings, then, depends on being able to breach the security of the commitment scheme. Since Pedersen Commitment is secure under the discrete logarithm assumption [99], so is SCULPT. A malicious party could choose to

1. Abort the protocol before any step,
2. publish $\perp$ instead of the correct bit-string, or
3. reveal a Corrupted bit-string in the *De-commitment Phase*.

Then, the protocol's response is:

Abort or $\perp$: The protocol is aborted and can either be restarted to compute the result correctly or aborted permanently.

Corrupted: The verification of the malicious party's (or parties') commitment(s) fails, unless the malicious party (or parties) is able to obtain the discrete logarithm between the generators h and g. Since this is computationally infeasible for the appropriate size of N, the malicious party (or parties) will be identified by the rest, should they choose to cheat.

Hence, none of these malicious actions can breach the privacy of the involved parties.

3.3 SECURE SHUFFLE AND EXPONENTIATION IN COVERT ADVERSARIES MODEL (SESAME)

SESAmE aims to enable verifiable shuffling and exponentiation of a set, when the input and output transcripts of the exponentiation are known to the *verifiers*. We require SESAmE to contain the following properties:

1. **Maintain privacy of set elements and secret key -** Neither the set elements exponentiated, nor the secret key used for exponentiation, should be leaked in the process of shuffling or exponentiation.
2. **Unlinkability between input and output -** An honest *verifier* should not be able to link individual elements between the input and output sets.
3. **Verifiability of shuffling and exponentiation -** It is provable, either under *covert* or *malicious adversaries* model, that all the elements in the output set are indeed elements from the input set, but shuffled, and exponentiated with the same value.

Research works [1, 43, 83, 94] focusing on verifiable shuffling, provide us some with the primitives to construct our protocol. However, proving correct shuffling requires number of exponentiations linearly dependent on the cardinality of the input set, in all the cited research. Hence, we have constructed a protocol assuming a relaxed security constraint, *covert adversaries* model, to achieve higher efficiency. Our protocol, SESAmE, in contrary to the

cited research, requires only 5 exponentiations to prove correct shuffling of input set, irrespective of the cardinality of the set.

SESAmE is secure in the *covert adversaries* model, where the maximum probability with which a party can cheat and escape is $\frac{1}{2}$. Protocol 6 depicts the design of SESAmE. In addition to the general setup, all parties are also privy to the Pedersen Commitment on the engaging party's secret key, the input set to be exponentiated, and a random bit-string whose length is equal to the cardinality of the input set, is disclosed to them at the beginning of the *Random Segregation Phase*. The protocol is divided into three phases, viz. *Exponentiation Phase*, *Random Segregation Phase* and *Proving Phase*. In the *Exponentiation Phase* the party engaged in the protocol exponentiates the input set elements with her secret key, shuffles, and posts the exponentiated set in the *broadcast channel*. The random bit-string is then learned and used to split the input and output sets into two subsets each, in the *Random Segregation Phase*. The product of the elements in each of the four newly formed subsets are then computed individually by all parties in the same phase. Finally, in the *Proving Phase*, the *prover* proves the discrete logarithm between the values computed in the previous phase is same as the secret key in the commitment over the *prover's* secret key.

The intuition behind the protocol is that, if the random bit-string is computed after the *Exponentiation Phase*, then no party can know which subset each element in the input set would belong to. Each item in the input set has equal chances of falling into either of the subsets. Hence, if a party were to cheat, she will be caught with non-negligible probabilities.

3.3.1 *Asymptotic complexity analyses*

Table 5 presents the step-wise asymptotic complexity analyses of the protocol. Different from the rest, only P_1 performs the proving and the remaining *n-1* parties are *verifiers*. The computation and communication complexities for the proving party, and thus overall, is $\mathcal{O}(k)$, linear with respect to the cardinality of the input set.

3.3.2 *Security analysis*

SESAmE is essentially a two-party protocol between a *Prover* and *Verifier*. We prove the security of SESAmE by considering the cases of a non-adaptive compu- tationally-bound adversary $\mathcal{A}$ corrupting the *prover*, and the *verifier*, separately. We use the standardized proof for *covert adversaries* model from [51] by including the security of *AND*-composition of zero-knowledge proofs as a primitive [40].

PROVER IS CORRUPTED In SESAmE, there is no threat to the privacy of the *verifier*. Thus, only the correctness of the protocol needs to be ensured from the viewpoint of the honest *verifier* when the *prover* is corrupted. We prove security, i.e. correctness in this case, by comparing the output distribution of computing

Protocol 6 : SESAmE

Additional Setup: P_1 engages in the shuffle and exponentiate protocol. All parties 1. are privy to $PC_{r_1}(K_1) = h^{K_1} g^{r_1}$, the Pedersen Commitment on P_1's secret key, 2. possess the input set S_n^1 to be exponentiated, and 3. become aware of a random bit-string R of length k, at the beginning of *Random Segregation Phase*.

1. **Exponentiation Phase -** P_1 computes and posts in the *broadcast channel* $S_n^2 = \Omega(\{S_n^1[j]^{K_1} |\ j \in [1,k]\})$, where Ω is a random permutation function.

2. **Random Segregation Phase -**

   a) P_1 creates two sets M_0 and M_1, where $M_\alpha = \{S_n^1[j] |\ R[j] = \alpha,\ \alpha \in \{0,1\}\}$.

   b) P_1 computes the sets N_0 and N_1, where $N_\alpha = \Omega(\{s^{K_1} |\ s \in M_\alpha\})$, where Ω is a random permutation function.

   c) P_1 computes g_0, g_1, y_0 and y_1, where $g_\alpha = \prod_s s,\ \forall s \in M_\alpha$ and $y_\alpha = \prod_s s,\ \forall s \in N_\alpha$.

   d) P_1 posts M_0, M_1, N_0, and N_1 in the *broadcast channel*.

3. **Proving Phase -** P_1 posts the Zero-Knowledge Proofs for $PC_{r_1}(K_1) = h^{K_1} g^{r_1}$, $y_0 = g_0^{K_1}$ and $y_1 = g_1^{K_1}$ in *AND*-composition to the *broadcast channel* using Fiat-Shamir Heuristics [40, 45].

SESAmE in the real model and the ideal model. Let the functionality f^{π_3} compute the result of SESAmE in the ideal model, and let Sim_3 be a probabilistic, polynomial time simulator. The function of Sim_3 in the ideal model is:

1. Sim_3 receives the input set S_n^1, exponentiated set S_n^2, and secret key K_1 from $\mathcal{A}$.

2. Sim_3 receives the bit-string R and zero-knowledge proofs from $\mathcal{A}$ and transmits it to the trusted party.

   a) *Case 1*: $\mathcal{A}$ exponentiated S_n^1 with a different key than K_1. The trusted party is able to detect the attempt to cheat, as the zero-knowledge proofs cannot be verified. The trusted party sends Corrupted to the Sim_3. Sim_3 then simulates the *prover* aborting and halts.

   b) *Case 2*: $\mathcal{A}$ exponentiated S_n^1 with K_1, but replaced a subset of k° ($<k$) elements T ($\subset S_n^2$) with a new, randomly generated set T' of the same cardinality. The cheating attempt is again detected by the trusted party, who sends Corrupted to the Sim_3. Sim_3 then simulates the *prover* aborting and halts.

   c) *Case 3*: $\mathcal{A}$ exponentiated S_n^1 with K_1, but replaced a subset of k° ($<k$) elements T ($\subset S_n^2$) with a new set T' of the same cardinality, s.t. $\prod_{\beta \in T} \beta = \prod_{\gamma \in T'} \gamma$.

		Computation Complexity		*Communication Complexity*	
		P_1	*Total*	P_1	*Total*
Step 1		k	k	k	k
Step 2	*a*	-	-	-	-
	b	-	-	-	-
	c	-	-	-	-
	d	-	-	$2k$	$2k$
Step 3		6	6	5	5
Overall		k+6	k+6	$3k$+5	$3k$+5

Table 5: Asymptotic complexity analyses of SESAmE

i. All the elements in the set T' fall into the same segregated, exponentiated set, i.e. either N_0 or N_1. The trusted party does not detect the cheating attempt as all zero-knowledge proofs can be verified. The trusted party sends $\mathrm{Undetected}$ to Sim_3. Sim_3 then halts.

ii. All the elements in the set T' do not fall into the same segregated, exponentiated set. The trusted party detects the cheating attempt as the zero-knowledge proofs cannot be verified. The trusted party sends $\mathrm{Corrupted}$ to the Sim_3. Sim_3 then simulates the *prover* aborting and halts.

d) *Case 4*: $\mathcal{A}$ exponentiated S_n^1 with K_1 correctly. Sim_3 simulates the behavior of a honest *prover* and then halts. We prove that,

$$\{\mathbf{IDEALSC}_{f^{\pi_3}, Sim_3}(S_n^1, K_1, R)\} \overset{\mathrm{C}}{\equiv} \{\mathbf{REAL}_{\mathsf{SESAme}, \mathcal{A}}(S_n^1, K_1, R)\}. \tag{27}$$

Since the behavior of the trusted party in the ideal model can be replicated by a honest *verifier* in the real model, Equation 27 holds. It is important to remember that in our security proof we do not assume that the trusted party chooses between $\mathrm{Corrupted}$ and $\mathrm{Undetected}$ based on a probability, unlike in [51]. Instead, we take the security of the *AND*-composition of zero-knowledge proofs as a given [40, 45], and expect the trusted party to be able to discern when the zero-knowledge proofs cannot be verified. A malicious *prover* can cheat without being detected only by following the strategy in *Case 3*. The probability that all k° fall into the same segregated, exponentiated set is $\frac{1}{2^{k^\circ-1}}$. When $k^\circ = 1$, there is no corruption in the output set. Hence, a malicious *prover* can cheat without being detected with a maximum probability of $\frac{1}{2^{(2-1)}} = \frac{1}{2}$, where $k^\circ = 2$.

VERIFIER IS CORRUPTED When the *verifier* is corrupted, the honest *prover* must ensure the privacy of the secret key K_1 and the random permutation function $\Omega()$. As in the previous case, we assume the security of the *AND*-composition of the zero-knowledge proof as a primitive [40, 45]. Hence, the adversary $\mathcal{A}$ should not be able to glean K_1 or $\Omega()$ from the input set, output set, and the intermediate transcripts. We should prove the

indistinguishability of the transcripts generated by the honest *prover* in the real model and simulated by a probabilistic polynomial time simulator Sim_4 in ideal model. Let the transcripts of Sim_4 be

$$Sim_4^{\pi_3}(S_n^1,\, K_1,\, r_1,\, PC_{r_1}(K_1)) = (\mathcal{S}_n^2,\, \mathcal{M}_0,\, \mathcal{M}_1,\, \mathcal{N}_0,\, \mathcal{N}_1), \tag{28}$$

where $\mathcal{S}_n^2$, $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{N}_0$, and $\mathcal{N}_1$ are randomly generated with the appropriate sizes by Sim_4. Let the transcripts of the honest *prover* in the real model be

$$Prover^{\pi_3}(S_n^1,\, K_1,\, r_1,\, PC_{r_1}(K_1)) = (S_n^2,\, M_0,\, M_1,\, N_0,\, N_1), \tag{29}$$

where S_n^2, M_0, M_1, N_0, and N_1 are correctly computed. Then, we can say that

$$Sim_4(S_n^1,\, K_1,\, r_1,\, PC_{r_1}(K_1)) \overset{\mathsf{C}}{\equiv} Prover^{\pi_3}(S_n^1,\, K_1,\, r_1,\, PC_{r_1}(K_1)), \tag{30}$$

under the discrete logarithm assumption [99]. Since zero-knowledge proofs are assumed to be secure and they do not reveal any information about the secret key K_1. Hence, SESAmE is secure against a malicious *verifier*.

While the indistinguishability of the elements before and after exponentiation is established using the discrete logarithm assumption [99], we are still required to prove that the protocol maintains their unlinkability from the viewpoint of the *verifier*. The input set, whose cardinality is k, is split into two sets, M_0 and M_1. Lets say that the cardinality of M_0 is $\breve{v}$, consequentially, the cardinality of M_1 is $k - \breve{v}$. The output set is similarly split into two sets, N_0 and N_1. These splits are performed such that N_α is the exponentiated and shuffled set of all the elements in M_α, for $\alpha \in \{0, 1\}$. Understandably, the cardinality of N_0 is $\breve{v}$ and that of N_1 is $k - \breve{v}$. Now, both N_0 and N_1 are shuffled secretly by the *prover*. Hence, to a verifier oblivious to the permutation functions, there are $\breve{v}\,!$ possible ways to permute N_0, and $(k - \breve{v})\,!$ possible ways to permute N_1. Hence, the likelihood, say $\mho$, of a *verifier* linking the input and output sets is

$$\mho(\breve{v}) = (\frac{1}{\breve{v}\,!}) \times (\frac{1}{(k - \breve{v})\,!}). \tag{31}$$

To find the optimum $\breve{v}$, we need to differentiate Equation 31 w.r.t $\breve{v}$, and equate the result to zero. However, it is not possible to differentiate the factorial function, hence we will approximate the factorial function with the Gamma function, s.t.

$$\Gamma(\breve{v} + 1) = \breve{v}\,!,\ \forall\, \breve{v} \in \mathbb{Z}_N, \tag{32}$$

where $\Gamma()$ is the Gamma function [34]. Now, differentiating Equation 31 w.r.t $\breve{v}$ we get,

$$\begin{aligned}\frac{d}{d\breve{v}}\mho(\breve{v}) = \frac{1}{(\breve{v}\,!) \times ((k-\breve{v})\,!)} \times (-\breve{\gamma} + \sum_{\breve{i}=1}^{k-\breve{v}}(\frac{1}{\breve{i}}))- \\ \frac{1}{(\breve{v}\,!) \times ((k-\breve{v})\,!)} \times (-\breve{\gamma} + \sum_{\breve{i}-1}^{\breve{v}}(\frac{1}{\breve{i}})) = 0 \\ \implies \sum_{\breve{i}=1}^{k-\breve{v}}(\frac{1}{\breve{i}}) - \sum_{\breve{i}-1}^{\breve{v}}(\frac{1}{\breve{i}}) = 0 \\ \implies (k-\breve{v}) - \breve{v} = 0 \\ \implies \breve{v} = k/2,\end{aligned} \tag{33}$$

as $\frac{d}{d\breve{v}}\Gamma(\breve{v}+1) = \breve{v}\,! \times (\breve{\gamma} + \sum_{\breve{i}=1}^{\breve{v}}(\frac{1}{\breve{i}}))$ [4], $\frac{1}{(\breve{v}\,!)\times((k-\breve{v})\,!)} \neq 0$ for $k \in \mathbb{Z}_N$, $\breve{v} \in [0,k]$, and where $\breve{\gamma}$ is the Euler–Mascheroni constant [4]. Since there is only a single optimum value, it is safe to consider this point a global optimum,but, we do know know if it is a global maximum or minimum. For $k = 20$, the optimum value for $\breve{v}$ is 10. For the same k, $\mho(0) = 4.11 \times 10^{-19}$, $\mho(10) = 7.60 \times 10^{-14}$, and $\mho(20) = 4.11 \times 10^{-19}$. We can see that $\mho(\breve{v})$ achieves its highest value for $\breve{v} = 10$, and hence, we can observe that $\breve{v} = k/2$ is, in fact, a global maximum for $\mho(\breve{v})$. Therefore, we can state that the chances of a *verifier* linking the input and output sets are maximized when $\breve{v}$ is set to $\frac{k}{2}$, and the upper bound in the probability of doing so is

$$\mho^{\circ} = (\frac{1}{\frac{k}{2}\,!})^2, \tag{34}$$

where $\mho^{\circ}$ is the maximum probability, for a given k. As we have shown, even for a value as low as $k = 20$, the maximum probability of linking the input and output sets is equal to 7.60×10^{-14} or 7.60×10^{-12} percent. This probability, is sufficiently low, especially considering it will only reduce further with increasing k.

4

PRIVATE SET UNION-COMBINATORIAL INTERSECTION CARDINALITY IN COVERT ADVERSARIES MODEL (SUCCINCT)

In this section, we describe and analyse our principal protocol, Private Set Union-Combinatorial Intersection Cardinality in Covert Adversaries Model (SUCCINCT). Similar to Protocol 4, this protocol aims to obtain the intersection cardinality between *client's* set and all union combination of sets held by a finite number of *vendors'*, without leaking the elements in each party's set to the rest. Different from the first protocol, this protocol is secure in the existence of *covert adversaries*. We choose the *covert adversaries* model instead of more robust *malicious adversaries* model to be able to construct efficient protocols that have practical security constraints. The functionality of SUCCINCT is, nonetheless, the same as SUCCINCT-HBC, and its result is $\mathcal{M}$ (as in Section 3.1).

SUCCINCT, detailed in Protocol 7, is comprised of five phases, viz. *Offline Phase*, *Posting Phase*, *Exponentiation Phase*, *Transmission Phase* and *Combinatorial Phase*. In contrast to Protocol 4, in the *Posting Phase*, each party posts a Pedersen Commitment to her secret key in addition to posting her elements exponentiated by her half key. Additionally, in the *Exponentiation Phase*, after each round of posting an exponentiated set, all parties engage in SCULPT protocol to generate the random bit-string required for each party to engage in *Step 2* and *Step 3* of SESAmE and publish the zero-knowledge proofs for the set that they exponentiated previously. Since all the parties jointly engage in SCULPT protocol after posting the exponentiated set, no party knows how the exponentiated set is to be segregated prior to posting it. Hence, any party can cheat without being detected with only a maximum probability of $\frac{1}{2}$.

It is important to note that the security in the protocol only prevents cheating when operating with other parties' sets. Thus, all parties are assumed to perform operations on their sets honestly. Since the protocol is likely to be followed with a set intersection protocol between the *client* and the chosen *vendors*, or with the *client* buying their services, it is safe to assume that any foul play by a *vendor* during SUCCINCT will be identified by the *client* at this juncture. Since the assumed application for this problem is to choose the appropriate *vendors*, it is reasonable to infer that any *vendor* caught cheating could lose the trust of other parties. The *vendors* involved in the protocol are CTI service providers and a loss of trust would affect their public profile, adversely affecting their business. This advantageous situation is also the reason why we secure our protocol in *covert adversaries* model instead of the more robust *malicious adversaries* model, as it enables us to leverage the social circumstances to devise more efficient solutions. Hence, assuming *covert adversaries* model in such cases is sufficient to get the parties to conform to the

Protocol 7 : SUCCINCT

1. **Offline Phase -** Each party $P_i,\ i \in [1,n]$
   a) chooses a secret key $K_i \in \mathbb{Z}^*_{\phi(N)}$.
   b) picks $K_i^L \in_R \mathbb{Z}^*_{\phi(N)}$ and then computes $\{K_i^R = K_i \times K_i^{L^{-1}} \mod \phi(N)\}$. Hence $K_i^L \times K_i^R = K_i \mod \phi(N)$.
2. **Posting Phase -** Each party $P_i,\ i \in [1,n]$,
   a) computes and posts $S_i^1 = \{(S_i[j])^{K_i^L} |\ j \in [1,k_i]\}$ in the broadcast channel.
   b) computes and posts the Pedersen's Comitment over their secret key, $PC_{r_i}(K_i) = h^{K_i} g^{r_i}$ after picking $r_i \in_R \mathbb{Z}^*_N$.
3. **Exponentiation Phase -** All parties engage in cyclic exponentiation of the posted elements lasting *n-1* rounds. In round $u,\ u \in [1, n-1]$,
   a) each set $S_i^u,\ i \in [1,n]$ is exponentiated by the party P_{i+u}, who creates and posts the new set $S_i^{u+1} = \Omega(\{(S_i^u[j])^{K_{i+u}} |\ j \in [1,k_i]\})$, where $\Omega()$ is a random permutation function.
   b) all parties engage in a SCULPT protocol to compute a random bit-string R_u of length t (where $t = \max_{i=1}^{n} k_i$).
   c) each party $P_{i+u},\ i \in [1,n]$ takes the first k_i bits from R_u as input for ***Step 2*** and ***Step 3*** of SESAmE protocol to compute and publish the appropriate zero-knowledge proofs, as the exponentiation is already performed by all parties prior to engaging in SCULPT.
4. **Transmission Phase -** Each party $P_i,\ i \in [1,n]$
   a) exponentiates the set S_i^n with her key K_i^R to obtain the set $Q_i = \{(S_i^n[j])^{K_i^R} |\ j \in [1,k_i]\}$.
   b) creates a Bloom filter representation of their set, BF_i, with k^* as the overall cardinality and a mutually accepted ϵ as the false positive rate, where $k^* = \sum_i k_i,\ i \in [1,n]$. Then transmits BF_i to P_1 in a secure channel. P_1 does not perform this action.
5. **Combinatorial Phase -** P_1 computes a membership matrix $\mathcal{M}$, s.t. $\mathcal{M}_i[j]$ represents the result of the membership test of element $Q_1[j]$ in the Bloom filter digest BF_i, $\forall\, i \in [2,n]$. The result of $\mathcal{M}_i[j]$ is *1* if the membership test is positive and *0* if negative.

protocol specifications.

SUCCINCT has the same drawback as SUCCINCT-HBC, in that the *client* can approximately estimate the intersection cardinality of two sets given their bloom filter digests [102]. This drawback does not reveal any information about the set

elements of the *vendors* to the client and the result is only an approximation. Nonetheless, we have addressed this issue in Section 5.2.

4.1 ASYMPTOTIC COMPLEXITY ANALYSES

Table 6 exhibits the step-wise analyses of the communication and computation requirements of the protocol. As in SUCCINCT-HBC, the most expensive phase is the *Exponentiation Phase*, with *n-1* rounds. Each iteration of SUCCINCT requires one iteration of SUCCINCT-HBC, *n-1* iterations of SCULPT, and *n(n-1)* iterations of SESAmE. However, the asymptotic complexities themselves remain unchanged between SUCCINCT-HBC and SUCCINCT ($Computation: \mathcal{O}(n^2k - nk\ln(\epsilon))$; $Communication: \mathcal{O}(n^2k(1 - \ln(\epsilon)))$).

		Computation Complexity		***Communication Complexity***	
		Each Party	*Total*	*Each Party*	*Total*
Step 1	*a*	-	-	-	-
	b	*1*	*n*	-	-
Step 2	*a*	*k*	*nk*	*k*	*nk*
	b	*1*	*n*	*1*	*n*
Step 3	*a*	*(n-1)k*	*n(n-1)k*	*(n-1)k*	*n(n-1)k*
	b	*2(n-1)*	*2n(n-1)*	*3(n-1)*	*3n(n-1)*
	c	*6(n-1)*	*6n(n-1)*	*(n-1)(2k+5)*	*n(n-1)(2k+5)*
Step 4	*a*	*k*	*nk*	-	-
	b	- ***or*** $k\ell^{\dagger}$	*k*ℓ*(n-1)*	- ***or*** $m^{\dagger}$	*(n-1)m*
Step 5		*k*ℓ ***or*** -†	*k*ℓ	-	-
Overall		*2(k*ℓ*+k+1)+ (n-1)(k+7)*	*2n(k*ℓ*+k+1)+ n(n-1)(k+7)*	*k+1+(n-1)(3k+8)* ***or*** *k+m+1+(n-1)(3k+8)*†	*n(k+m+1)+ n(n-1)(3k+8)-m*

Table 6: Asymptotic complexity analyses of SUCCINCT

† *Estimation for Client* **or** *Estimation for Vendors*

4.2 SECURITY ANALYSIS

We prove the security of SUCCINCT using the composition theorem of [24]. We use the security of SUCCINCT-HBC, SCULPT, and SESAmE as primitives in the composition. The *Exponentiation Phase* in SUCCINCT is essentially *n(n-1)* iterations of SESAmE, where the random bit-string R is jointly computed by all parties in each round of the *Exponentiation Phase* using SCULPT. Hence, using the composition theorem for *malicious adversaries* [24] and *covert adversaries* [9], we can state that the *Exponentiation Phase* in SUCCINCT is secure under the *covert adversaries* model, as it is the more relaxed security assumption. In the other phases of SUCCINCT, all parties only handle their own set. If a party P_i incorrectly computes S_i^1 or BF_i, it will not be detected by the other parties. However, no party can corrupt her set in a way that she can deterministically

manipulate the result of the protocol, due to the random permutation functions used. Hence, by corrupting her set, a *vendor* only stands to corrupt the result of protocol in an unforeseeable way, and the *client* only stands to corrupt the results, which are exclusively revealed to her. Thus, no party is incentivized to mishandle her own set.

The *client* cannot easily link the elements in her original set (S_1) and her final exponentiated set (Q_1), due to the unlinkability property of SESAmE we use in the *Exponentiation Phase*. Since the phase has $n-1$ rounds of shuffling and exponentiation, the maximum likelihood of the *client* establishing the link is equal to

$$(\frac{1}{\frac{k}{2}!})^{2(n-1)}. \tag{35}$$

Equation 35 can be derived from Equation 34 by considering the number of rounds of SESAmE employed in SUCCINCT per set. For just 10 participants, with 10 elements each, in SUCCINCT, this likelihood is equal to 8.4×10^{-119}. Hence, we can state that SUCCINCT sufficiently unlinks S_1 and Q_1. The *vendors* do not obtain the Bloom filter digests of the other parties, thus the unlinkability analysis need not be performed from their viewpoint.

5

EXTENSIONS

In this section, we discuss two modifications that we apply to SUCCINCT. The first to improve SUCCINCT's privacy at the cost of increased computation overhead, and the second to decrease the asymptotic complexities at the cost of decreased security. The two modifications, SUCCINCT-O and SUCCINCT-d, will be discussed and analyzed in the following subsections, along with the primitives necessary to understand the modifications.

5.1 PRIMITIVES

MULTI-PARTY KEY AGREEMENT PROTOCOL As the name suggests, such protocols aim to jointly establish a key among a number of parties. The properties we require of the protocol are,

1. **Security against malicious participants -** The key agreement protocol has to be secure against *malicious adversaries*, especially to ensure that no malicious participant is able to prevent all involved parties from computing the same key.
2. **Not share the joint key in plaintext -** The joint key must only be known to the participants of the protocol, and not be shared in plaintext at any point in the protocol.
3. **Passive adversary resistant -** No passive adversary, with access to all intermediate transcripts, is capable of computing the same secret key as the participants of the protocol.

The key agreement protocol that fits our above mentioned requirements is that by Tseng [104]. Tseng's protocol is described in Protocol 8. All the participants are connected in a circular structure, as shown in Figure 7. The computation and communication complexities of $\pi_{MKAP-MA}$ are $\mathcal{O}(\eta^2)$ and $\mathcal{O}(\eta)$, respectively, for η participants in the protocol. While there exists protocols for multi-party key agreement [16, 56, 101] that have linear asymptotic complexities, they are not secure in the presence of *malicious adversaries*. To check if a participant Γ_τ, $\tau \in [1, \eta]$, has cheated, any party involved in $\pi_{MKAP-MA}$ will just have to verify if two equations are valid, which are,

$$\begin{aligned} \lambda^{\gamma_\tau} &= v_\tau \times \phi_\tau^{\mathcal{H}(\omega_\tau, v_\tau, \iota_\tau)} \mod \Lambda \\ \left(\frac{\phi_{\tau+1}}{\phi_{\tau-1}}\right)^{\gamma_\tau} &= \iota_\tau \times \omega_\tau^{\mathcal{H}(\omega_\tau, v_\tau, \iota_\tau)} \mod \Lambda \end{aligned} \tag{36}$$

Protocol 8 : $\pi_{MKAP-MA}$[104]

Initial setup: The protocol consists of η parties, say $\Gamma_1, \ldots, \Gamma_\eta$. All parties are aware of the values λ, Ξ, and Λ. Ξ, and Λ are large primes with $\Lambda = 2 \times \Xi + 1$, and λ is a generator for the subgroup G_Ξ, s.t. $G_\Xi = \{\theta^2 | \theta \in \mathbb{Z}_\Lambda^*\}$. In addition all parties are also aware of a cryptographic hash function $\mathcal{H}$.

1. **Step 1 -** Each party $\Gamma_\tau,\ \tau \in [1, \eta]$,
   a) generates her secret share $\zeta_\tau \in \mathbb{Z}_\Xi$.
   b) broadcasts $\phi_\tau = \lambda^{\zeta_\tau} \mod \Lambda$ to all parties.
2. **Step 2 -** Upon receiving $\phi_\tau,\ \forall\, \tau \in [1, \eta]$, each party $\Gamma_\tau,\ \tau \in [1, \eta]$,
   a) generates $\beta_\tau \in_R \mathbb{Z}_\Xi$
   b) computes and published ω_τ, v_τ, ω_τ, and ω_τ, s.t.,

$$\begin{aligned} \omega_\tau &= \left(\frac{\phi_{\tau+1}}{\phi_{\tau-1}}\right)^{\zeta_\tau} \mod \Lambda \\ v_\tau &= \lambda^{\beta_\tau} \mod \Lambda \\ \iota_\tau &= \left(\frac{\phi_{\tau+1}}{\phi_{\tau-1}}\right)^{\beta_\tau} \mod \Lambda \\ \gamma_\tau &= \beta_\tau + \zeta_\tau \times \mathcal{H}(\omega_\tau, v_\tau, \iota_\tau) \mod \Xi \end{aligned} \tag{37}$$

3. **Step 3 -** Each party $\Gamma_\tau,\ \tau \in [1, \eta]$, can compute the secret key Y using the formula,

$$\begin{aligned} \mathrm{Y} &= \phi_{\tau-1}^{\eta \times \zeta_\tau} \times \omega_\tau^{\eta-1} \times \omega_{\tau+1}^{\eta-2} \times \ldots \times \omega_{\tau-1} \mod \Lambda \\ &= \lambda^{\zeta_1\zeta_2 + \zeta_2\zeta_3 + \zeta_3\zeta_4 + \ldots + \zeta_{\tau-1}\zeta_\tau} \mod \Lambda \end{aligned} \tag{38}$$

5.2 OUTPUT OBFUSCATION (SUCCINCT-O)

One of the limitations of SUCCINCT is that the *client* can share the Bloom filter representations received from the *vendors* with any one of them. Obtaining these representations would enable any *vendor*, involved in SUCCINCT, to compute the set union-combinatorial intersection cardinalities between her set and the sets held by other *vendors*. To prevent this, we are required to obfuscate the Bloom filters transmitted by the *vendors* in such a way that only the *client* can discern useful information from them. To this end we modify the *Transmission Phase* and *Combinatorial Phase* in Protocol 7. The modified phases are explained in Protocol 9. The only two phases that are modified from SUCCINCT, in SUCCINCT-O, are *Transmission Phase* and *Combinatorial Phase*. In the *Transmission Phase*, all parties by the *client* commit to another secret key, say K_i' for each party P_i, which is then used to exponentiate their own set resulting from the *Exponentiation Phase*, on top of their other *half*-keys. Additionally, all *vendors* also exponentiate S_1^n (the final exponentiated set of the *client* from the *Exponentiation Phase*) with their second secret keys, which they transmit to the

client along with the appropriate zero-knowledge proofs to prove their exponentiations. Finally, in the *Combinatorial Phase*, the *client* exponentiates all the sets received from the *vendors* in the *Transmission Phase* with her *half*-key, to be able to query the Bloom filter representations and compute the membership matrix $\mathcal{M}$. The intuition behind the modification is that, adding another layer of exponentiation on top of SUCCINCT renders each party's set indistinguishably random to all other parties, except for the *client*. The *client* is the only party whose set is exponentiated by the second secret key of the *vendors*, and thus, is the only party that can discern information from the Bloom filter representations. Additionally, since different Bloom filters contain elements exponentiated with different values, the *client* cannot estimate the intersection cardinality of two sets given their Bloom filter digests.

Protocol 9 : SUCCINCT-O

---

4. **Transmission Phase -** Each party $P_i,\ i \in [2, n]$
   a) generates another secret key $K_i' \in \mathbb{Z}^*_{\phi(N)}$ and posts a Pedersen Commitment, $PC_{r_i'}(K_i') = h^{K_i'} g^{r_i'}$ on it into the *broadcast channel*, after picking $r_i' \in_R \mathbb{Z}^*_N$.
   b) exponentiates the set S_i^n with $K_i^R \times K_i'$ to obtain the set $W_i = \{(S_i^n[j])^{K_i^R \times K_i'} |\ j \in [1, k_i]\}$.
   c) exponentiates all elements in S_1^n with K_i' to obtain the set $S_{1i}^n = \{(S_1^n[j])^{K_i'} |\ j \in [1, k_i]\}$, and transmits S_{1i}^n to P_1 along with the zero-knowledge proofs for exponentiating S_1^n and $PC_{r_i'}(K_i')$ in *AND*-composition using Fiat-Shamir Heuristics.
   d) creates a Bloom filter representation of their exponentiated set W_i, BF_i', with k^* as the overall cardinality and a mutually accepted ϵ as the false positive rate, where $k^* = \sum_i k_i,\ i \in [1, n]$. Then transmits BF_i' to P_1 in a secure channel.
5. **Combinatorial Phase -** P_1 exponentiates all sets $S_{1i}^n,\ i \in [2, n]$, with K_1^R to obtain $Q_{1i} = \{(S_{1i}^n[j])^{K_1^R} |\ j \in [1, k_1]\},\ i \in [2, n]$. P_1, then, computes a membership matrix $\mathcal{M}$, s.t. $\mathcal{M}_i[j]$ represents the result of the membership test of element $Q_1[j]$ in the Bloom filter digest BF_i, $\forall\, i \in [2, n]$. The result of $\mathcal{M}_i[j]$ is *1* if the membership test is positive and *o* if negative.

Since the resulting Bloom filter representations in SUCCINCT-O does not contain elements exponentiated by the same value, we can no longer obtain the lossless union combinations from the Bloom filter representations. However, that is not required as the the membership matrix $\mathcal{M}$ can still be computed. This computation is possible only due to the fact that no party permutes the set S_1^n in the *Transmission Phase*. Hence, the j^{th} element in the sets $Q_{1i},\ i \in [2, n]$, all correspond to the same element in S_1.

5.2.1 *Asymptotic Complexity Analysis*

As stated previously, the modification suggested in SUCCINCT-O increases the computation overhead as compared to SUCCINCT. The asymptotic complexities of SUCCINCT-O, however, is the same as SUCCINCT (*Computation* : $\mathcal{O}(n^2k - nk\ln(\epsilon))$; *Communication* : $\mathcal{O}(n^2k(1 - \ln(\epsilon)))$). Table 7 presents the asymptotic complexities of SUCCINCT-O, with the steps differing from SUCCINCT marked with $*$.

		Computation Complexity		***Communication Complexity***	
		Each Party	*Total*	*Each Party*	*Total*
Step 1	*a*	-	-	-	-
	b	*1*	*n*	-	-
Step 2	*a*	*k*	*nk*	*k*	*nk*
	b	*1*	*n*	*1*	*n*
Step 3	*a*	*(n-1)k*	*n(n-1)k*	*(n-1)k*	*n(n-1)k*
	b	*2(n-1)*	*2n(n-1)*	*3(n-1)*	*3n(n-1)*
	c	*5(n-1)*	*5n(n-1)*	*(n-1)(2k+5)*	*n(n-1)(2k+5)*
Step 4*	*a*	- ***or*** *2*†	*2(n-1)*	- ***or*** *1*†	*(n-1)*
	b	- ***or*** *k*†	*(n-1)k*	-	-
	c	- ***or*** *(2k+2)*†	*(n-1)(2k+2)*	- ***or*** *(2k+4)*†	*(n-1)(2k+4)*
	d	- ***or*** $k\ell^\dagger$	$k\ell(n-1)$	- ***or*** $m^\dagger$	*(n-1)m*
Step 5*		*(n-1)k(ℓ+1)* ***or*** -†	*(n-1)k(ℓ+1)*	-	-
Overall*		*(n-1)(2k+7)+* *(n-1)kℓ+(k+2)* ***or*** *(4k+6)+k* *+(n-1)(k+7)ℓ*†	$n^2(k+7)+$ $n(4k-1)-$ $4(k+1)+$ $2(n-1)k\ell$	*(n-1)(3k+8)+(k+1)* ***or*** *(3k+6+m)+* *(n-1)(3k+8)*†	$n^2(3k+8)+$ $n(m-2)-$ $(2k+m+5)$

Table 7: Asymptotic complexity analyses of SUCCINCT-O

† *Estimation for Client* **or** *Estimation for Vendors*

5.2.2 *Security Analysis*

The security of SUCCINCT-O is derived from the proof for SUCCINCT. The difference in SUCCINCT-O is that there is another round of exponentiation, to ensure that all sets are exponentiated with different keys. This modification ensures the security of SUCCINCT-O, even when there is collusion between the *client* and a group of *vendors*. Since SUCCINCT-O differs from SUCCINCT only in the last two phases, we just prove that the new intermediate transcripts in SUCCINCT-O are computationally indistinguishable from random values to anyone who did not generate them.

From the viewpoint of a *vendor*, say P_i, the additional transcripts due to the modification are r'_i, K'_i, $PC_{r'_i}(K'_i)$, S^n_{1i}, W_i, and BF'_i. The zero-knowledge proofs are assumed to be secure and taken as a primitive [40, 45]. Of the new transcripts, $PC_{r'_i}(K'_i)$ is visible to all involved parties, and S^n_{1i} and BF'_i are just visible to the *client*. Let Sim_5 be a probabilistic, polynomial-time simulator. Sim_5 randomly generates $\mathcal{PC}_{r'_i}(K'_i)$, $\mathcal{S}^n_{1i}$, $\mathcal{BF}'_i$ of the appropriate sizes and structures. We show that,

$$\{\mathcal{PC}_{r'_i}(K'_i),\, \mathcal{S}^n_{1i},\, \mathcal{BF}'_i\} \overset{c}{\equiv} \{PC_{r'_i}(K'_i),\, S^n_{1i},\, BF'_i\}. \tag{39}$$

Equation 39 is valid, i.e. the randomly generated transcripts are computationally indistinguishable from the actual transcripts to all parties, except P_i, under the discrete logarithm assumption [99]. Hence, no party can discern any information from the visible transcripts other than the party who generated them. The security proof from the viewpoint of the *client* is unchanged from SUCCINCT, as there are no additional transcripts generated by her. Therefore, SUCCINCT-O is secure under the *covert adversaries* model, as is SUCCINCT. Additionally, since each *vendor's* set is exponentiated with a different value, no information is leaked even if the *client* were to share the Bloom filter representations with other parties.

5.3 d-OUT-OF-n KEY SHARING (SUCCINCT-D)

The joint secret key K is shared among all n parties in SUCCINCT. This requirement can be relaxed by splitting the n parties into $\frac{n}{d}$ groups, s.t. $n \mod d = 0$, and evaluating SUCCINCT per group. The final exponentiated sets of all parties, however, must be exponentiated with the same key. Hence, this splitting has to be achieved by utilizing a key agreement protocol to establish a key among $\frac{n}{d}$ parties, d times. This key agreement algorithm has to be secure in the presence of *malicious adversaries*, not leak the secret key, and not share the secret key in plaintext during the protocol. To this end, we use $\pi_{MKAP-MA}$ described in Protocol 8. The original set of parties $\mathcal{P}$ is split into d subsets, s.t.

$$\mathcal{P}_b = \{P_i | i \in [b, b + \frac{n}{d}]\},\ b \in [0, d),\ b \in \mathbb{W}. \tag{40}$$

Each group $\mathcal{P}_b$, $b \in [0, d)$, engages in $\pi_{MKAP-MA}$ to jointly compute a secret key K^b, which will be used as the key K_i by each party $P_i \in \mathcal{P}_b$. The original set of parties $\mathcal{P}$ is again split into $\frac{n}{d}$ subsets of d parties each, s.t.

$$\mathcal{P}^v = \{\mathcal{P}_b[v] | b \in [0, d),\ b \in \mathbb{W}\},\ v \in [1, \frac{n}{d}]. \tag{41}$$

Each set $\mathcal{P}^v$, $v \in [1, \frac{n}{d}]$, then engages in SUCCINCT, and all parties at, the end of their groups' iteration of SUCCINCT, send their Bloom filter representations to the *client* (P_1). Thus, this modification, SUCCINCT-d, just involves $\frac{n}{d}$ iterations of SUCCINCT, with d parties each, and d iterations of $\pi_{MKAP-MA}$, with $\frac{n}{d}$ parties each.

SUCCINCT-d has a relaxed notion of security than SUCCINCT. While each iteration of SUCCINCT in SUCCINCT-d is secure in the *covert adversaries* model,

the overall secret key K in SUCCINCT-d is only computed from d shares, while the secret key in SUCCINCT is split amongst all involved parties.

5.3.1 *Asymptotic Complexities Analysis*

The asymptotic complexities of SUCCINCT-d can be computed from the individual asymptotic complexities of SUCCINCT and $\pi_{MKAP-MA}$. In SUCCINCT-d, the individual iterations of SUCCINCT needs an extra step, which is, all parties must send their Bloom Filter representations to the *client*. Adding this extra step, however, does not change the asymptotic complexities in comparison to the original SUCCINCT. Hence the computation complexity of SUCCINCT-d is,

$$\frac{n}{d} \times \mathcal{O}(d^2k + dk\ln(\epsilon)) + d \times \mathcal{O}\left(\left(\frac{n}{d}\right)^2\right) = \mathcal{O}\left(ndk + dk\ln(\epsilon) + \frac{n^2}{d}\right), \quad (42)$$

where $\mathcal{O}(d^2k)$ and $\mathcal{O}((\frac{n}{d})^2)$ represent the computation complexity of SUCCINCT and $\pi_{MKAP-MA}$, respectively, for the assumed initial constraints. Similarly, the communication complexity of SUCCINCT-d is,

$$\frac{n}{d} \times \mathcal{O}(d^2k(1 - \ln\epsilon)) + d \times \mathcal{O}\left(\frac{n}{d}\right) = \mathcal{O}(ndk(1 - \ln\epsilon) + n), \quad (43)$$

where $\mathcal{O}(d^2k(1 - \ln\epsilon))$ and $\mathcal{O}(\frac{n}{d})$ represent the communication complexity of SUCCINCT and $\pi_{MKAP-MA}$, respectively, and ϵ is the false positive rate agreed upon for the Bloom filters. The asymptotic complexities of SUCCINCT-d, therefore, are $\mathcal{O}(ndk + dk\ln(\epsilon) + \frac{n^2}{d})$ for computation, and $\mathcal{O}(ndk(1 - \ln\epsilon) + n)$ for communication.

5.3.2 *Security Analysis*

We prove the security of SUCCINCT-d using the composition theorem of [24]. SUCCINCT is secure in the *covert adversaries* model, and $\pi_{MKAP-MA}$ is secure in the *malicious adversaries* model. Hence, using the composition theorem for *malicious adversaries* [24] and *covert adversaries* [9], we can state that SUCCINCT-d is secure under the *covert adversaries* model, as it is the more relaxed security assumption. It is important to remember that, for an adversary to retrieve the secret joint key in SUCCINCT-d, she would just have to compromise d appropriate participants. The same adversary would have to compromise all n participants in SUCCINCT.

6

OPTIMIZATION FOR MAXIMUM OVERLAP (COMBO)

To achieve our research goal, we are required to identify a subset with the smallest possible number of *vendors*, while maximizing the intersection cardinality between the *client's* set and the union set of the subset of *vendors*. Computing all union combinations to identify the required one, however, is not efficient as there are $2^{n-1} - n$ combinations. That would imply that the computation complexity of evaluating the required result is then at least $\mathcal{O}(2^n)$. The required result can, nonetheless, be efficiently extracted from the membership matrix $\mathcal{M}$ using Protocol 10. The mechanism employed by Protocol 10 is simple. A list of *vendors*, $\mathfrak{V}$, is first created, where the *vendors* are sorted in ascending order, based on their intersection cardinality with the *client's* set. This can be accomplished by using a comparison based sorting scheme, and modifying the scheme to compare the intersection cardinalities of the individual *vendors'* sets and the *client's* set instead. In cOMbO, the sorting scheme proposed by Pok-Son and Arne [65] is used to sort the list of *vendors*. This sorting scheme is chosen as it has a worst-case computation complexity of $\mathcal{O}(n \log n)$, for n items in the list, which is the best among comparison based sorting schemes. The *vendor(s)* whose set has the least intersection cardinality with the *client's* set would be first in the list $\mathfrak{V}$, and those with the maximum intersection cardinality would be last in the same list. The *client* then computes $S_{\mathfrak{V}} = \cup_{\forall P_i \in \mathfrak{V}}$. Finally, the *client* iterates through the list $\mathfrak{V}$ and removes every *vendor* P_i from $\mathfrak{V}$, for whom

$$|S_1 \cap S_{\mathfrak{V}}| = \left|S_1 \cap S_{\mathfrak{V}-\{P_i\}}\right|. \tag{44}$$

All the remaining vendors in the set $\mathfrak{V}$ is a smallest subset of *vendors* whose union set has the maximum intersection cardinality with S_1, among all different possible subsets of *vendors*. We purposefully mention "a smallest subset" in the previous sentence as there is a possibility that number of different subsets of *vendors*, with the same number of *vendors* in the subset, can have maximum intersection cardinality with the *client's* set. This reason is why we sort the list of *vendors* based on their sets' intersection cardinality with the *client's* set. Consequentially, the redundant *vendors* with relatively low overlap with the *client's* set are removed first. By considering both the individual and combined cardinality of the vendors, the *client* ensures that she chooses a best combination of *vendors*, while also maximizing information obtained from individual *vendors* simultaneously.

6.1 EFFICACY ANALYSIS

cOMbO will always result in a subset of *vendors*, that has the least possible number of *vendors*, whose union set has the maximum intersection cardinality with the *client's* set, among all possible subset of *vendors*. If this solution is

Protocol 10 : cOMbO

Additional Setup:List of *vendors* $\mathfrak{V}$, sorted in ascending order based on their intersection cardinality with the *client's* set.

```
Function GetUnionCard(M, V*)
    Data : Membership matrix M and a sorted sublist of vendors V*
    Result : |S_1 ∩ S_V*|
    Card = 0;
    for j in [1, k_1] do
        temp = False;
        for P_v in V* do
            if M_v[j] == 1 then
                temp = True
            end
            if temp == True then
                Card = Card + 1
            end
        end
    end
    return Card
end

for P_v in V do
    if (GetUnionCard(M, V) == GetUnionCard(M, V − {P_v})) then
        V = V − {P_v}
    end
end
```

unique, cOMbO will obtain it. To prove the efficacy of cOMbO, let us consider a sorted list of vendors $\mathfrak{V} = [P_2, \ldots, P_n]$. Now, let us assume that we are in the process of iteration and are evaluating the efficacy of *vendor* P_i. There are two possible outcomes, which are

1. $\boldsymbol{|S_1 \cap S_{\mathfrak{V}}| = |S_1 \cap S_{\mathfrak{V}-\{P_i\}}|}-$ This implies that P_i is redundant, i.e P_i does not contribute any set elements that is not already present in $S_1 \cap S_{\mathfrak{V}-\{P_i\}}$, and thus, P_i can be remove from $\mathfrak{V}$. Additionally, the other *vendors* in $\mathfrak{V}$ all have either higher or the same intersection cardinality with the client as P_i. This implies that while we progressively remove redundant *vendors*, we take care to remove the ones with lest relevance to the *client* first.

2. $\boldsymbol{|S_1 \cap S_{\mathfrak{V}}| > |S_1 \cap S_{\mathfrak{V}-\{P_i\}}|}-$ This implies that P_i contains element(s) that is not present in $S_1 \cap S_{\mathfrak{V}-\{P_i\}}$. Hence, P_i is essential and must not be removed.

From these two actions it can be understood that for the resulting subset, say $\mathfrak{V}^\star$, $|S_1 \cap S_{\mathfrak{V}^\star}|$ is always equal to $|S_1 \cap S_{\mathfrak{V}}|$. Hence, the resulting subset $\mathfrak{V}^\star$ results in the maximum possible intersection cardinality with the *client's* set, and is of the smallest possible size, as none of the *vendors* in $\mathfrak{V}^\star$ are redundant.

The computation complexity of cOMbO can be computed from the computation complexity of the sorting algorithm used and the function $\mathrm{GetUnionCard}()$. Let the cardinality of the *client's* set be $k_1 = k$. Their computation complexities are:

- **Sorting Algorithm -** As mentioned previously, the sorting protocol used in cOMbO has a computation complexity of $\mathcal{O}(n \log n)$ for n items in the list to be sorted. This implies that there are $\mathcal{O}(n \log n)$ comparisons required. Since the compared values in our sorting algorithm is the intersection cardinality between the sets held by individual *vendors*, in the list, and the *client's* set. Consequentially, the sorting algorithm required $\mathcal{O}(n \log n)$ intersection cardinality computations. These individual intersection cardinalities can be easily computed from the membership matrix $\mathcal{M}$. For a *vendor* P_i with the set S_i,

$$|S_1 \cap S_i| = \sum_{j=1}^{k} \mathcal{M}_i[j], \tag{45}$$

  where S_1 is the *client's* set, and k is its cardinality. Evaluating Equation 45 required iterating over k items, and, naturally, has a computation complexity of $\mathcal{O}(k)$. Therefore, the computation complexity of our sorting algorithm is $\mathcal{O}(n \log n) \times \mathcal{O}(k) = \mathcal{O}(kn \log n)$.

- $\mathrm{GetUnionCard}()$ **-** The function performs maximum of 4 operations inside the confines of a nested loop structure. Let the two loops iterate over k and, say, n^* respectively. Then, the maximum number of operations performed for a single iteration of $\mathrm{GetUnionCard}()$ is $4 \times k \times n^*$, and thus its computation complexity is $\mathcal{O}(n^* k)$.

cOMbO has the worst-case computation complexity when the size of the list $\mathfrak{V}$ is maximized in each call to $\mathrm{GetUnionCard}()$. This case is due to the linear dependence of $\mathrm{GetUnionCard}()$'s computation complexity on the size of the list $\mathfrak{V}$. This implies that none of the vendors must be removed from $\mathfrak{V}$. Consequentially, as cOMbO iterates through $\mathfrak{V}$, each iteration has two function calls to $\mathrm{GetUnionCard}()$, and there are totally $n-1$ such iterations (since $n-1$ *vendors* are expected to be involved). Among these two calls, one function call transmits the sorted list as argument with $n-1$ *vendors*, and the other with $n-2$ *vendors*. Therefore, the worst-case overall complexity of cOMbO is equal to

$$\mathcal{O}(kn \log n) + (n-1) \times \mathcal{O}((n-1)k) + (n-1) \times \mathcal{O}((n-2)k) = \mathcal{O}(n^2 k). \tag{46}$$

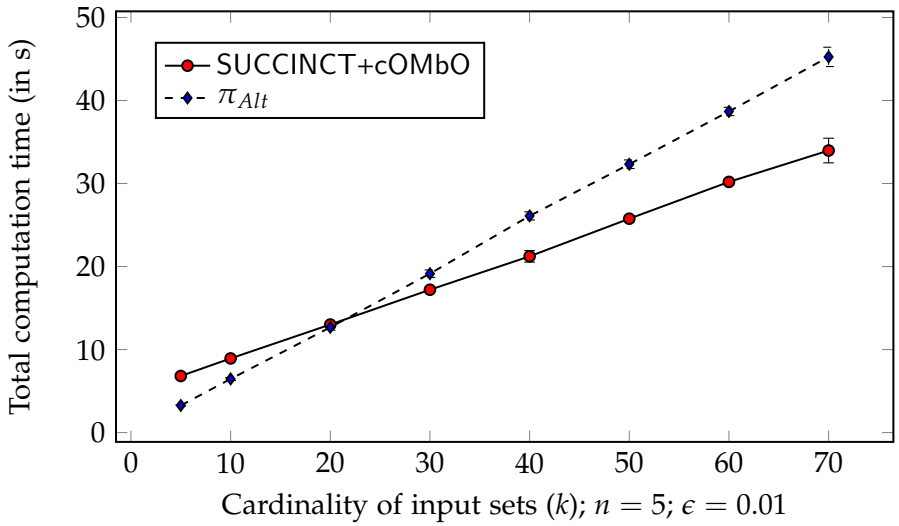
It is important to note that the computation complexity of cOMbO is $\mathcal{O}(n^2 k)$ "plain-text" operations, which does not involve computationally expensive operations such as modular exponentiation and hashing.

7

IMPLEMENTATION AND VALIDATION

In this section, we analyze the real-time performance of our principal protocol SUCCINCT (in conjuncture with (cOMbO)) by comparing it to the most efficient current alternative. We created naive implementations of both approaches in Python 2. We run the experiments on a machine which runs Ubuntu 16.04LTS operating system on Intel (R) Xeon (R) E5-2643 64-bits, where *RAM:* 125 GB; *Speed:* 3.30 GHz. To meet the current security standards [13], we choose a 2048 bit prime number as the prime module of operations.

As discussed in Section 2.9, the most efficient alternative to SUCCINCT is constructed using the two-party set intersection cardinality protocol from De Cristofaro et al. [29] and the multi-party set union protocol from Blanton and Aguiar [19]. Our implementation of the most efficient alternative protocol (π_{Alt}), however, does not include the multi-party set union protocol of [19] as we are unable to implement some of the primitives included in their work. Instead, we implement the private set intersection cardinality protocol by [29] in conjuncture with a simple, non-private set union protocol. We assume that an arbitrary party in each combination engages in the private set intersection with the *client*, representing that combination. We measure the computation time of our experiments by averaging over 10 executions for each value of n and k, where n is the number of parties and k is the cardinality of their sets. The average computation times for the two approaches, under different initial constraints, are shown in Figure 8.



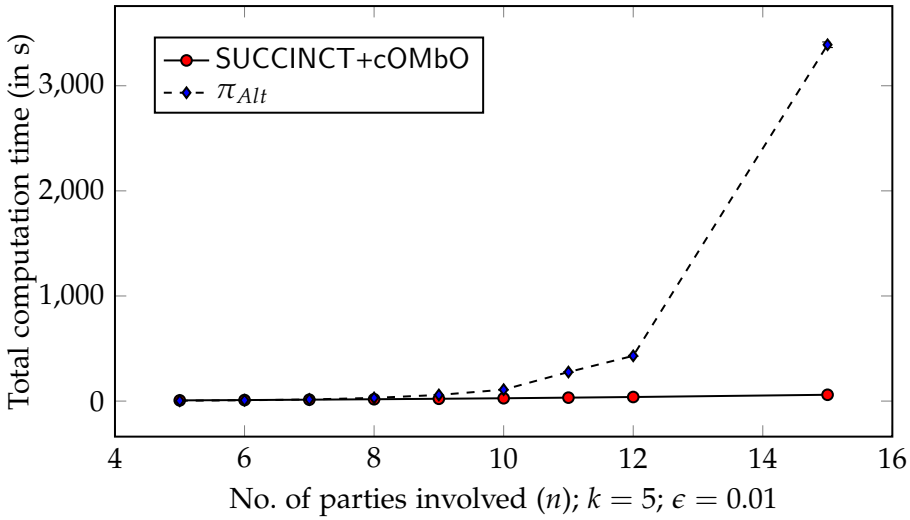
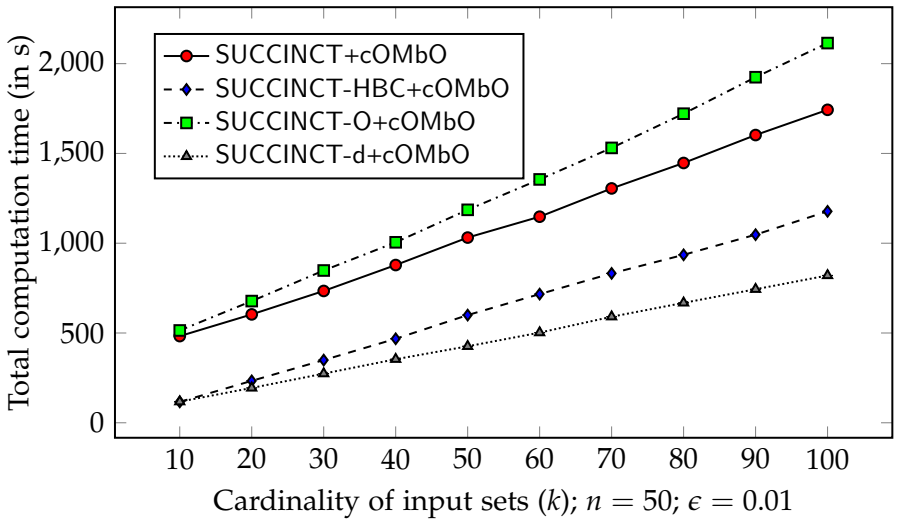


Figure 8: Performance comparison of SUCCINCT and π_{Alt}

The comparison in Figure 8 clearly shows that our protocol outperforms π_{Alt}. The values of n and k are kept low as the implementation of π_{Alt} is infeasible for large n. Both SUCCINCT's and π_{Alt}'s performances are linearly dependent on k, with π_{Alt} having a larger coefficient. Moreover, the difference in the computation time between the two approaches widens as the number of involved parties increase. This is because, the implemented protocol resulting from the most efficient alternative approach suffers a computation complexity of at least $\mathcal{O}(2^n k)$, while ours is only $\mathcal{O}(n^2 k - nk \ln \epsilon)$. The performance of the individual protocols is hard to gauge for varying n, due to the huge difference in computation time between the two. SUCCINCT computes the result under the mentioned initial conditions in 1 minute, while π_{Alt} requires 56.5 minutes for the same. The implementation for the most efficient alternative would be computationally more expensive if the multi-party set union protocol by Blanton and Aguiar [19] was included.



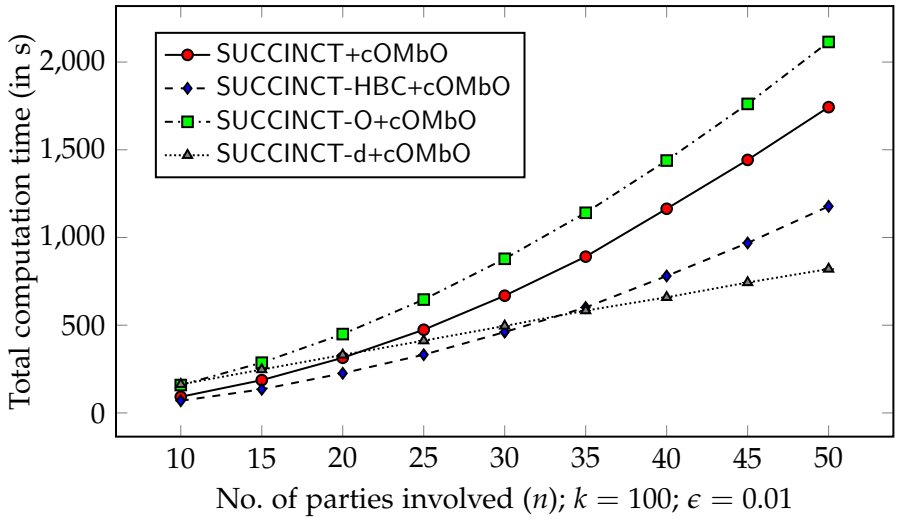
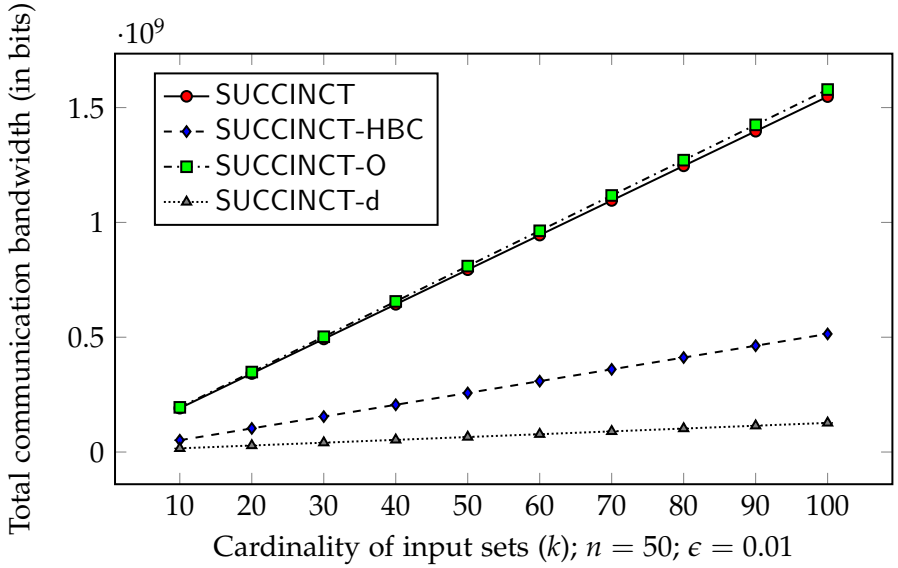


Figure 9: Performance comparison of SUCCINCT, SUCCINCT-HBC, SUCCINCT-O, and SUCCINCT-d

We also compare the real-time performance of SUCCINCT-HBC, SUCCINCT, SUCCINCT-O, and SUCCINCT-d (all combined with cOMbO). The performance comparison between naive implementations of our protocols is shown in Figure 9. The comparison was performed under the same hardware and software constraints as the previous performance comparison. The results were averaged over 3 iterations and the standard deviations are shown as error bars around the data points. π_{Alt} is not included in this comparison, as the difference in performance between π_{Alt} and our protocols is too high for large values of n. The difference makes it hard to discern the differences in performance between our protocols. It is important to note that SUCCINCT-d starts to outperform all the other protocols for large values of n, and appears to have a linear dependence on the number of involved participants, atleast for



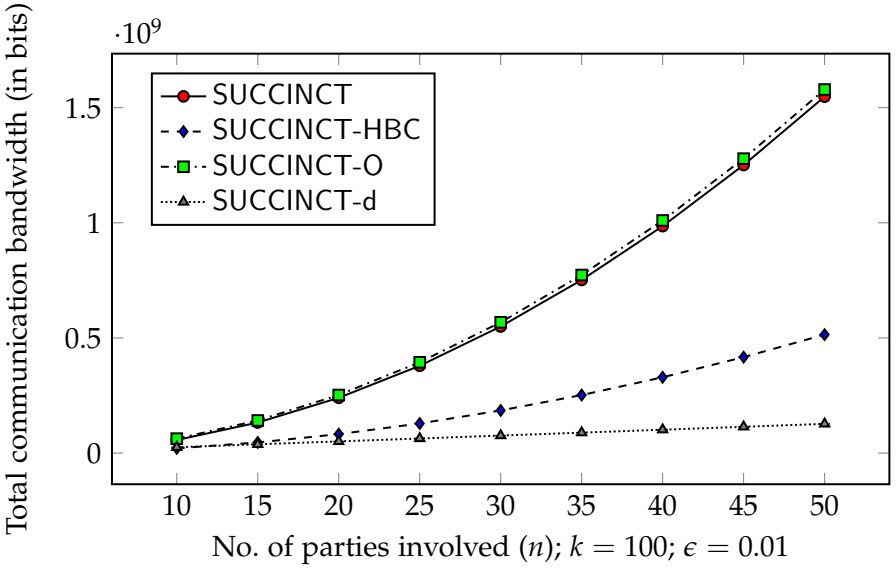


Figure 10: Communication bandwidth comparison of SUCCINCT, SUCCINCT-HBC, SUCCINCT-O, and SUCCINCT-d

the chosen range of the number of participants.

The communication bandwidth required of our protocols for the same initial settings as Figure 9, is shown in Figure 10. The bandwidth is estimated from the communication complexity analyses of our protocols rather than from their naive implementations. The bandwidth required do not change for different iterations of our protocols, as long as the number of participants, the number of items in their sets, and the Bloom filter false positive rate does not vary. Hence, we are not required to measure them from the protocol implementations.

The use of Bloom filters introduces a probabilistic nature to our protocols. There is room for some error when computing the intersection cardinalities

ϵ	SUCCINCT	SUCCINCT-HBC	SUCCINCT-O	SUCCINCT-d (d=5)
10^{-2}	1.54×10^9	5.14×10^8	1.57×10^9	1.26×10^8
10^{-4}	1.55×10^9	5.16×10^8	1.58×10^9	1.27×10^8
10^{-8}	1.55×10^9	5.21×10^8	1.58×10^9	1.27×10^8
10^{-16}	1.56×10^9	5.30×10^8	1.59×10^9	1.28×10^8
10^{-32}	1.58×10^9	5.49×10^8	1.61×10^9	1.29×10^8
% change in bandwidth	2.28	6.85	2.23	2.27

Table 8: Bandwidth (in bits) with respect to false positive rate ϵ ($n = 50, k = 100$)

due to the possibility of having false positives when querying Bloom filters. However, the false positive rate of Bloom filters is an attribute that can be preset. Moreover, this attribute can be set very low without discernibly affecting the bandwidth of communication. The reduction in false positive rate has negligible influence in the computation performance of our protocols, as it only increases the number of hashing operations, whose computation costs are insignificant in comparison to the exponentiation operation. The impact of the false positive rate on the communication bandwidth of our protocols is assessed in Table 8.

Table 8 also shows the percentage change in the bandwidth of our protocols, for decreasing the false positive rate from $\epsilon = 10^{-2}$ to $\epsilon = 10^{-32}$. The maximum increase in the bandwidth is noticed in SUCCINCT-HBC, which is around 6.85%. We can infer from this statistic that reducing the false positive rate has a negligible impact on the bandwidth of our protocols. This implication can also be recognized from the fact that the communication complexity of our protocols is only linearly dependent on the natural logarithm of the false positive rate of Bloom filters. Hence, we can construe that the false positive rate in our protocols can be set very low, such that the likelihood of error is virtually eliminated, without having a perceivably adverse impact on their communication bandwidth.

8

DISCUSSION AND FUTURE WORK

The rise in the amount and impact of cyber attacks [79] has required organizations to actively improve their defensive posture from a purely passive one (acting after the compromise) to a more active one (predict, prevent, detect, respond) [63]. Because cyber security is often not the core business of a company, it is often perceived as only an ancillary function. To invest in cyber security countermeasures both efficiently and effectively, maximizing the return over investment is necessary. One way this is done in practice is by studying the tactics, techniques and procedures specific to threat actors. This process usually is performed by collecting what are called indicators of compromise (IOCs), which are atomical information like hashes, IPs, etc. Since a plethora of CTI sources exist, and usually not for free, organizations should select their provider by using the relative relevance of their indicators databases as a criterion. In this research, we aim to construct protocols to enable organizations to find their appropriate CTI vendors by focusing on the following research question:

> *"How to devise a protocol to enable a party (Client) with a set of items, chosen from a finite universe, to find a subset of other parties (Vendors) with similar sets, in such a way that the union set, of all vendors in the said subset, has the maximum set intersection cardinality with the Client's set, while maintaining the privacy of all participants' set elements?"*

In this chapter, we return to the proposed research question and discuss how the protocols presented achieve our research goal. Additionally, we identify avenues for future research by recognizing open problems and improvements.

8.1 DISCUSSION

In this thesis, we presented efficient protocols to compute union-combinatorial intersection cardinality in *honest-but-curious* and *covert adversaries* models. The protocols are based on modular exponentiation and Bloom filters. The first protocol, **SUCCINCT-HBC**, is secure in the *honest-but-curious adversaries* model. The protocol preserves the privacy of the involved parties in the presence of *malicious adversaries*, however it is still possible to compromise the correctness of protocol execution. To this end, we created two additional preliminary protocols to improve the security of SUCCINCT-HBC, viz. SCULPT, and SESAme. **SCULPT** is a simple coin-tossing protocol to jointly compute a random bit-string among multiple, mutually-mistrusting parties, in *malicious adversaries* model. We modified the simple two-party coin-tossing protocol by Lindell [70] to obtain SCULPT, as per our requirement. **SESAmE** is a protocol

for secure shuffling and exponentiation in the presence of *covert adversaries*. An important feature of SESAmE is that we condense the number of zero-knowledge proofs to three, no matter how many items are exponentiated. Also, by splitting the items in the input set into two groups, we maintain the unlinkability between the sets before and after exponentiation, in spite of providing security in the presence of *covert adversaries*. We obtain our principal **SUCCINCT** by combining SUCCINCT-HBC, SCULPT, and SESAmE. SUCCINCT obtains the same output as SUCCINCT-HBC, but is also robust to the presence of *covert adversaries*. SUCCINCT enjoys the same asymptotic complexities of SUCCINCT-HBC, however, it is more expensive from the viewpoint of real-time implementation.

In SUCCINCT the security is valid as long as our assumption, that there is no collusion between the *client* and *vendors*, holds. Any other *vendor* can obtain the set union-combinatorial intersection cardinality between her set and the sets held by other *vendors* by obtaining the final Bloom filter representations from the *client*. Additionally, in SUCCINCT-HBC and SUCCINCT the *client* can obtain estimations of intersection cardinalities between different *vendors*, without compromising the privacy of the actual set elements. To circumvent these issues, we propose our first modification, **SUCCINCT-O**, to SUCCINCT, by adding an extra layer of modular exponentiation in the *Transmission Phase* of SUCCINCT. SUCCINCT-O, thus, obfuscates the Bloom filters in such a way that only the *client* can discern any information from them. SUCCINCT-O enjoys the same asymptotic complexities of SUCCINCT, although the real-time performance is more expensive in comparison. Our second modification **SUCCINCT-d** aims to improve on the asymptotic complexities of SUCCINCT, while still maintaining security in the *covert adversaries* model. In SUCCINCT-d, we segregate the participants into groups, and evaluate SUCCINCT per group. By using $\pi_{MKAP-MA}$, we ensure that all the items in all the Bloom filter representations are exponentiated by the same value.

In summary, all our four protocols, SUCCINCT, SUCCINCT-HBC, SUCCINCT-O, and SUCCINCT-d, evaluate the same functionality albeit with different security constraints. SUCCINCT-HBC is appropriate for applications where the participants of the protocol can be trusted to follow the protocol description, while still ensuring the privacy of their set elements. SUCCINCT restricts the participants from deviating from protocol specifications by leveraging on their social image, although under the assumption of no collusion between *client* and *vendors*. SUCCINCT-O boasts the same security as SUCCINCT, while additionally being robust to the previously mentioned collusion. SUCCINCT-d allows for manipulating the asymptotic complexities of the protocol by splitting the joint secret key into fewer pre-determinable number of shares. From their asymptotic complexity and performance analysis, we can discern that, for any value of n and k, SUCCINCT-HBC outperforms SUCCINCT, which in turn outperforms SUCCINCT-O. The security of the three protocols, understandably, is inversely proportional to their performance. SUCCINCT-d, on the other hand, behaves as if it is linearly dependent on the

number of participants, and thus will always outperform the other three protocols for large values of n. SUCCINCT-d is, thus, particularly useful for a large number of participants when the joint key does not necessarily have to be split between all of them.

SUCCINCT-HBC, SUCCINCT, SUCCINCT-O, and SUCCINCT-d all provide the union-combinatorial intersection cardinality between the *client's* set and the *vendors'* sets in the form of a membership matrix $\mathcal{M}$. For any given combination of *vendors*, denoted δ, $|S_1 \cap S_\delta|$ can be easily computed from $\mathcal{M}$, where S_δ is the union set of the sets held by the *vendors* in δ. While the required subset can be naively obtained by computing the intersection cardinality of all union-combinations of *vendors'* sets and comparing them, such an approach would have to compute and compare $2^{n-1} - 1$ combinations of *vendors*, for $n - 1$ *vendors*. The result format $\mathcal{M}$ is specifically chosen as it allows for optimally extracting the required subset. Thus, the task of obtaining the best required subset of *vendors* is, thus, shifted to efficiently extracting the result from $\mathcal{M}$. **cOMbO** provides an efficient alternative to identify the subset of *vendors*, whose union set has the maximum overlap with the *client's* set. cOMbO achieves this result with just $\mathcal{O}(n^2k)$ plain-text operations, in the worst-case.

In Section 1.3, we recognized four sub-questions to our research question. In the first sub-question, we explore how to prevent sharing privacy-leaking transcripts during the protocol execution. As mentioned in Section 2.9, the most efficient current alternative requires a two-step process, where the first step is to evaluate the multi-party set union and the second step is to evaluate the intersection cardinality between the *client's* set and the computed union set. This approach leaks more information, about the set elements of the *vendors*, than what is revealed as the result of the protocol. We prevent the presence of such intermediate transcripts by splitting each party's key into two. This approach ensures that all the elements, published into the *broadcast channel* at the end of *Exponentiation Phase*, are not exponentiated by the same value. Should they all be exponentiated by the same value, all the involved parties can compare the exponentiated sets of all parties, revealing the intersection cardinalities between all parties' sets. Since each party posts their set exponentiated by a randomly generated half-key in the *Posting Phase*, the published sets at the end of *Exponentiation Phase* appear random until they are exponentiated with the other half-key. This half-key approach ensures that SUCCINCT-HBC, SUCCINCT, SUCCINCT-O, and SUCCINCT-d prevent sharing of privacy-leaking transcripts.

In the second sub-question, we inquire how to make our protocol robust to collusion between malicious *client* and *vendors*. SUCCINCT-HBC, and SUCCINCT are not robust to such collusions. To this end, we designed SUCCINCT-O. SUCCINCT-O includes an additional layer of exponentiation to obfuscate the final exponentiated sets of all parties. This modification, however, prevents all items in the protocol from being exponentiated with the same value. Hence, to mitigate this issue, each *vendor* also exponentiates the *client's*

exponentiated set. The additional layer of exponentiation prevents other *vendors* from gleaning any information from the Bloom filter representations, even if the *client* were to share it with them. While this modification increases the overall number of exponentiations by all parties, it leaves the asymptotic complexities unchanged as compared to SUCCINCT.

The third sub-question focuses on how the social obligations of the involved parties can be used to devise an efficient protocol. Our application setting aims to enable a *client* to chose her appropriate set of *vendors*. The *vendors* in our application are CTI service providers, who stand to incur losses in the form of public embarrassment, or loss of reputation, if they were to be caught cheating a *client*. We use this situation to our advantage by designing protocols secure in the *covert adversaries* model. The *covert adversaries* model allows for the participant of the protocol to cheat, but any cheating party will be caught with a non-negligible probability. In our protocols secure in the *covert adversaries* model, the maximum probability with which a participant can cheat without getting caught is $\frac{1}{2}$. Assuming the *covert adversaries* model enables us to reduce the total number of zero-knowledge proofs required per participant during each round of the *Exponentiation Phase* to just three, irrespective of how many elements there in the set to be exponentiated. Moreover, assuming security in *malicious adversaries* model, would require us to have zero-knowledge proofs to prove the correct exponentiation of each item in the set to be exponentiated. This assumption will not only adversely affect the computation and communication overhead of our protocol, but provide traceability between the set before and after exponentiation. We are required to prevent this traceability to ensure that the *client* cannot deduce any more information about the *vendors'* sets than the intersection cardinality between their and her set. Hence, our assumption of *covert adversaries* model not only improves the communication and computation overhead, but also prevents compromising on the *vendors'* privacy in our protocols.

The fourth and final sub-question examines if our protocols are computationally superior to the most efficient current alternatives in real-time performance. To this end, we compared of the naive implementations of SUCCINCT and the protocol constructed from the most efficient current alternatives π_{Alt}. SUCCINCT significantly outperforms π_{Alt} and the difference in performance only increases with increase in the number of participants and/or the number of items in each participant's set. For *15* participants with *5* items each, π_{Alt} required an overall runtime of *56.5 minutes*, while SUCCINCT

	SUCCINCT	SUCCINCT-HBC	SUCCINCT-O	SUCCINCT-d (d=5)
Runtime (in mins)	29.04	19.63	35.24	13.67

Table 9: Runtime analysis of our protocols ($n = 50$; $k = 100$)

required just *1* minute for the same initial conditions. A runtime analysis, of just our protocols, for *50* participants with *100* items each, is shown in Table 9. All of our protocols, involving *50* participants with *100* items each, outperform π_{Alt}, involving *15* participants with *5* items each.

8.2 FUTURE WORK

The protocols presented in this work are the first, to the best of our knowledge, to compute set union-combinatorial intersection cardinalities in *honest-but-curious* and *covert adversaries* model. All the protocols constructed in this thesis are computationally superior to the most efficient current alternative to compute the same, while simultaneously improving on the privacy of the involved participants. However, in spite of significantly improving on π_{Alt}, there is opportunity for improving on the work presented in this thesis.

OTHER OPTIMAL APPROACHES cOMbO efficiently identifies a smallest subset of *vendors*, whose union set has the maximum intersection cardinality with the *client's* set, among all possible subsets. It is, however, not the only optimization that can be employed. cOMbO sorts the list of all *vendors*, in ascending order based on their individual intersection cardinality with the *client*, and proceeds to eliminate them iteratively. In cOMbO the elimination happens when the *vendor* under consideration is redundant, thus, any *vendor* who contributes a unique element to the union set will not be eliminated. This elimination criteria could be altered to obtain the best subset of *vendors* based on other requirements. For instance, Protocol 11 (π_{CMO}) presents how to alter cOMbO to include the cost of *vendors'* service, say C_i for vendor P_i, and a threshold value, say $\check{\gamma}$. π_{CMO}, in contrast to cOMbO, eliminates a *vendor*, P_i, only if the value added by P_i is greater than a preset threshold $\check{\gamma}$. This value added is computed by normalizing the number of unique elements, added to the union thanks to P_i, with the cost of its service, C_i. By doing so, the *client* can remove *vendors*, whose cost, from the *client's* view point, outweighs their added value.

Protocol 11 : π_{CMO}

Additional Setup:List of *vendors* $\mathfrak{V}$, sorted in ascending order based on their intersection cardinality with the *client's* set.

for P_v ***in*** $\mathfrak{V}$ **do**
  if $\left(\frac{(\texttt{GetUnionCard}(\mathcal{M},\mathfrak{V})-\texttt{GetUnionCard}(\mathcal{M},\mathfrak{V}-\{P_v\}))}{C_v}\right) \leq \check{\gamma}$ **then**
    $\mathfrak{V} = \mathfrak{V} - \{P_v\}$
  end
end

A number of other optimization algorithms can also be devised based on different requirements. Additional work is required to explore what other

requirements can be achieved using optimization algorithms, given the result format $\mathcal{M}$.

USAGE OF UNDENIABLE CRYPTOGRAPHIC ACCUMULATORS SUCCINCT, SUCCINCT-O, and SUCCINCT-d only ensures that any party cheating during the *Exponentiation Phase* is caught with a non-negligible probability. Nothing prevents a party from incorrectly computing transcripts during the *Posting Phase* and the *Transmission Phase*. Hence, the protocol does not ensure that each party handles her own sets appropriately. In our application setting, any *vendor* wrongly computing the Bloom filter representation corrupts the results in only a non-deterministic way. Moreover, a cheating *vendor* is also likely to suffer negative consequences if any evidence of cheating is found out. Hence, we leverage the social circumstances of the involved participants to mitigate this issue. However, there are one-way accumulators, that we can employ in place of Bloom filters, that allow for provable security, at the cost of higher computation overhead.

Lipmaa [72] formally defined this required property as *undeniability*. According to that definition, a cryptographic accumulator is *undeniable* if it is computationally infeasible to prove the membership and non-membership test of the same value, whether that value is a part of the accumulator or not. It is important to note that, according to the definition, the cryptographic accumulator has to be *universal* first prior to considering its *undeniability*. A *universal* accumulator is one that allows for verifiable membership and non-membership tests for any values. There are extant research works, such as [8, 21, 22, 33, 36], that have already achieved *undeniability* in their *universal* accumulators. [8, 33, 36] use primitives, such as commitments, and zero-knowledge proofs, while [21, 22] use sorted hash trees [76]. As such, they allow for incorporating provable security into the *Transmission Phase* of our protocols. Future works could explore the changes in utility and performance, resulting from integrating such cryptographic accumulators into our protocols.

IMPROVE THE PROBABILITY TO CATCH CHEATERS SUCCINCT, SUCCINCT-O, and SUCCINCT-d all incorporate security in the *Exponentiation Phase*, by utilizing SESAmE to verifiably shuffle and exponentiate sets. The maximum probability with which any party can cheat, in our protocols, and escape is $\frac{1}{2}$, as shown in Section 3.3.2. This can be reduced further by modifying the length of the random bit-string (t) in the *Random Segregation Phase* of SESAmE (Protocol 6). If t is made to be equal to twice the cardinality of the set to be exponentiated (k), we can use use two bits instead of one, per item in the set, to split the set. Likewise, by setting $t = \Phi \times k$, for a preset value Φ, we can allocate Φ bits per item in the set. By setting t so, we can split the set, before and after exponentiation, into 2^{Φ} subsets. Under this splitting scenario, let us assume the *prover* replaces k° items from the set after shuffling and exponentiating, such that the product of the replaced items is equal to that

of the replacing items. In this case, the *prover* escapes getting caught if all the items fall into a single subset, the likelihood of which is equal to

$$(\frac{1}{2^{\Phi}})^{k^{\circ}} \times 2^{\Phi} = \frac{1}{2^{\Phi(k^{\circ}-1)}}. \tag{47}$$

Equation 47 is maximized by setting $k^{\circ} = 2$, which results in the maximum probability, with which a cheater can escape getting caught, being equal to $\frac{1}{2^{\Phi}}$. However, if the value of Φ is too high, the protocols might leak additional information to the *client*. The leakage is due to the fact that, for high values of Φ the number of items in each subset, after splitting, becomes too low, allowing the *client* to link the items in her initial set and final exponentiated set. The linkage might reveal information about the actual items in the *vendors* sets, instead of just the intersection cardinality between sets. Hence, additional work is required to research the optimal values for Φ, such that it strikes the required balance between reducing the probability of successfully cheating, and privacy leakage resulting from it.

QUANTITATIVE TO QUALITATIVE COMPARISON The research goal of this thesis is to allow for a *client* to choose the cyber threat intelligence *vendors* appropriate to her, by using sets of items held by all participants. This comparison, however, is purely quantitative. The result of the protocols, suggested in this thesis, only informs the *client* about the intersection cardinalities between her set and all union-combinatorial groupings of *vendors'* sets. The *client* does not learn any information about the quality of the services provided by each *vendor* as a result of our protocols. This limitation shows an avenue to improve our work, and incorporate qualitative analysis of the *vendors'* services in choosing the appropriate *vendors*. Some of the research cited [7, 30, 41, 61], from the field of PPSO, have considered the problem of transferring relevant data along with the elements in the input sets.

Incorporating data transfer could also help validate that the low-level IOCs encountered by the *client* are not maliciously generated by a *vendor*. Such a scenario could result in the *client* identifying this *vendor* as more pertinent, than she actually is, due to the artificially generated IOCs. Although, our protocols also leverage on the public image of the *vendors* to conform them to the protocol description, this added feature could improve *client's* trust on the efficacy of the protocol. Incorporating this feature into our protocols might be problematic, due to the shuffling component in them. Nonetheless, the gap identified warrants additional research.

HIDING CORRELATION BETWEEN VENDORS In this thesis, the goal is to achieve privacy of the set elements through the protocol, which we have achieved. However, the membership matrix $\mathcal{M}$ does reveal the correlation between the different *vendors* involved in the protocol. The problem could be solved by using just the cOMbO protocol, and primitives for multi-party set union cardinality and two-party set intersection cardinality protocols. This is because cOMbO iteratively makes repeated function calls to $\mathrm{GetUnionCard}()$. Instead of deriving the result from the membership matrix $\mathcal{M}$, the result of $\mathrm{GetUnionCard}()$ could be derived from instantiating a multi-party set union

cardinality protocol along with a two-party set intersection cardinality protocol. Additionally the initial sorted list of *vendors* can also be achieved with two-party set intersection cardinality protocols between the *client* and each *vendor*. The resulting protocol would eliminate the need to construct the membership matrix $\mathcal{M}$. While this construction results in reduced fuctionality and flexibility, as compared to our protocols, its asymptotic complexities, performance and security warrants future research due to the possible improvement in security that we stand to make.

8.3 CONCLUDING REMARKS

The research goal of this thesis is to allow for organizations to choose the CTI vendors appropriate to them, by using sets of items held by all participants. While this comparison is purely quantitative, it is a step in enabling organizations to make informed decisions and optimize their expenditure in cyber security. Protocols constructed from the existing works, to compute this functionality, are infeasible to implement due to their exponential dependence on the number of participants involved. Additionally, such protocols contain privacy-leaking intermediate transcripts.

In this thesis, we present efficient protocols to compute *union-combinatorial intersection cardinality* in *honest-but-curious* and *covert adversaries* models. The protocols are based on modular exponentiations and Bloom filters. All the proposed protocols, SUCCINCT-HBC, SUCCINCT, SUCCINCT-O, and SUCCINCT-d, avoid generation of privacy-compromising intermediate transcripts. Our work also included two additional preliminary protocols: one to compute a random bit-string among multiple parties in *malicious adversaries* model and one to verifiably shuffle and exponentiate a set of elements in *covert adversaries* model. Our shuffle and exponentiate protocol boasts asymptotic complexities linear to the cardinality of the set to be exponentiated. The naive implementation and comparison of SUCCINCT and π_{Alt}, proves that our protocol is significantly faster than the most efficient current alternative. Further corroborated by the comparison of our protocols, involving more participants with a larger number of elements than the previous comparison.

While there certainly are limitations in our goal, the results from our research prove that this work could be a first step towards making the optimized use of CTI ubiquitous. With the burgeoning landscape of attacks on and through the digital infrastructure, along with our increasing dependence on this infrastructure, we have a dire need to formulate a proactive defense strategy. To formulate such a strategy, we need to envision an infrastructure which enables organizations to make monetarily optimized decisions, in choosing from a variety of CTI service providers, appropriate to them. Our work enables organizations to quantitatively choose relevant service providers using intel collected from past adversarial attacks, or reconnaissance attempts. Additional research to eliminate or, at the very least, mitigate our limitations, could prove significant in the materialisation of our envisioned infrastructure.

BIBLIOGRAPHY

[1] Masayuki Abe and Fumitaka Hoshino. "Remarks on Mix-Network Based on Permutation Networks." In: *Public Key Cryptography*. Vol. 1992. Lecture Notes in Computer Science. Springer, 2001, pp. 317–324.

[2] William Aiello, Yuval Ishai, and Omer Reingold. "Priced Oblivious Transfer: How to Sell Digital Goods." In: *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*. 2001, pp. 119–135.

[3] Mark Allman, Ethan Blanton, Vern Paxson, and Scott Shenker. "Fighting Coordinated Attackers with Cross-Organizational Information Sharing." In: (2006).

[4] Emil Artin and Michael Butler. *The gamma function*. Dover books on mathematics. New York, NY: Dover, 2015.

[5] Gilad Asharov and Yehuda Lindell. "A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation." In: vol. 18. 2011, p. 36.

[6] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. "More efficient oblivious transfer and extensions for faster secure computation." In: *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. 2013, pp. 535–548.

[7] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. "(If) Size Matters: Size-Hiding Private Set Intersection." In: *IACR Cryptology ePrint Archive* 2010 (2010), p. 220.

[8] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. "Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems." In: *CT-RSA*. Vol. 5473. Lecture Notes in Computer Science. Springer, 2009, pp. 295–308.

[9] Yonatan Aumann and Yehuda Lindell. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries." In: *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*. 2007, pp. 137–156.

[10] Yossi Azar, Andrei Z Broder, Anna R Karlin, and Eli Upfal. "Balanced allocations." In: *SIAM journal on computing* 29.1 (1999), pp. 180–200.

[11] Tom Ball. *Top 5 critical infrastructure cyber attacks*. https://www.cbronline.com/cybersecurity/top-5-infrastructure-hacks/. (Accessed on 12/27/2017). July 2017.

[12] George Bamford, John Felker, and Troy Mattern. "Operational Levels of Cyber Intelligence." In: *INSA, 2013* (2013).

[13] Elaine Barker and Quynh Dang. *NIST Special Publication 800–57 Part 1, Revision 4*. Tech. rep. NIST, 2016.

[14] Sean Barnum. "Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX™)." In: *MITRE Corporation* 11 (2012), pp. 1–22.

[15] Kenneth E. Batcher. "Sorting Networks and Their Applications." In: *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968*. 1968, pp. 307–314.

[16] Klaus Becker and Uta Wille. "Communication Complexity of Group Key Distribution." In: *ACM Conference on Computer and Communications Security*. ACM, 1998, pp. 1–6.

[17] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)." In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 1988, pp. 1–10.

[18] David Bianco. *The Pyramid of Pain*. 2014. URL: http://detect-respond.blogspot.nl/2013/03/the-pyramid-of-pain.html.

[19] Marina Blanton and Everaldo Aguiar. "Private and oblivious set and multiset operations." In: *Int. J. Inf. Sec.* 15.5 (2016), pp. 493–518.

[20] Burton H. Bloom. "Space/Time Trade-offs in Hash Coding with Allowable Errors." In: *Commun. ACM* 13.7 (1970), pp. 422–426.

[21] Ahto Buldas, Peeter Laud, and Helger Lipmaa. "Accountable certificate management using undeniable attestations." In: *ACM Conference on Computer and Communications Security*. ACM, 2000, pp. 9–17.

[22] Ahto Buldas, Peeter Laud, and Helger Lipmaa. "Eliminating Counterevidence with Applications to Accountable Certificate Management." In: *Journal of Computer Security* 10.3 (2002), pp. 273–296.

[23] Jan Camenisch and Victor Shoup. "Practical Verifiable Encryption and Decryption of Discrete Logarithms." In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. 2003, pp. 126–144.

[24] Ran Canetti. "Security and Composition of Cryptographic Protocols: A Tutorial." In: *IACR Cryptology ePrint Archive* 2006 (2006), p. 465.

[25] Manuel Castells. *The Impact of the Internet on Society: A Global Perspective*. https://www.technologyreview.com/s/530566/the-impact-of-the-internet-on-society-a-global-perspective/. (Accessed on 12/26/2017). Sept. 2014.

[26] Octavian Catrina and Sebastiaan de Hoogh. "Improved Primitives for Secure Multiparty Integer Computation." In: *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*. 2010, pp. 182–199.

[27] Louis Columbus. *The State Of The Subscription Economy, 2018*.

[28] Danielle Correa. *Cyber-criminals reap the benefits of cross-community collaboration*. https://www.scmagazineuk.com/cyber-criminals-reap-the-benefits-of-cross-community-collaboration/article/651890/. Apr. 2017.

[29] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. "Fast and Private Computation of Cardinality of Set Intersection and Union." In: *Cryptology and Network Security, 11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings*. 2012, pp. 218–231.

[30] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. "Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model." In: *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*. 2010, pp. 213–231.

[31] Emiliano De Cristofaro and Gene Tsudik. "Practical Private Set Intersection Protocols with Linear Complexity." In: *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*. 2010, pp. 143–159.

[32] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[33] Ivan Damgård and Nikos Triandopoulos. "Supporting Non-membership Proofs with Bilinear-map Accumulators." In: *IACR Cryptology ePrint Archive* 2008 (2008), p. 538.

[34] Philip J. Davis. "Leonhard Euler's Integral: A Historical Profile of the Gamma Function: In Memoriam: Milton Abramowitz." In: *The American Mathematical Monthly* 66.10 (1959), pp. 849–869. ISSN: 00029890, 19300972. URL: http://www.jstor.org/stable/2309786.

[35] Kristen Dennesen, John Felker, Tonya Feyes, and Sean Kern. "Strategic Cyber Intelligence." In: *INSA, 2014* (2014).

[36] David Derler, Christian Hanser, and Daniel Slamanig. *Revisiting Cryptographic Accumulators, Additional Properties and Relations to other Primitives*. Cryptology ePrint Archive, Report 2015/087. https://eprint.iacr.org/2015/087. 2015.

[37] Changyu Dong, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol." In: *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. 2013, pp. 789–800.

[38] Ulfar Erlingsson, Mark Manasse, and Frank McSherry. "A cool and practical alternative to traditional hash tables." In: *Proc. 7th Workshop on Distributed Data and Structures (WDAS'06)*. 2006.

[39] Shimon Even, Oded Goldreich, and Abraham Lempel. "A Randomized Protocol for Signing Contracts." In: (1982), pp. 205–210.

[40] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*. 1986, pp. 186–194.

[41] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient private matching and set intersection." In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2004, pp. 1–19.

[42] Keith B. Frikken. "Privacy-Preserving Set Union." In: *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*. 2007, pp. 237–252.

[43] Jun Furukawa and Kazue Sako. "An Efficient Scheme for Proving a Shuffle." In: *CRYPTO*. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 368–387.

[44] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms." In: *IEEE Trans. Information Theory* 31.4 (1985), pp. 469–472.

[45] Oded Goldreich and Hugo Krawczyk. "On the Composition of Zero-Knowledge Proof Systems." In: *SIAM J. Comput.* 25.1 (1996), pp. 169–192.

[46] Oded Goldreich and Rafail Ostrovsky. "Software Protection and Simulation on Oblivious RAMs." In: *J. ACM* 43.3 (1996), pp. 431–473.

[47] Michael T. Goodrich. "Randomized Shellsort: A Simple Oblivious Sorting Algorithm." In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*. 2010, pp. 1262–1277.

[48] Geoff Hancock, Christian Anthony, and Lincoln Kaffenberger. "Tactical Cyber Intelligence." In: *INSA, 2014* (2014).

[49] Reg Harnish. *What It Means To Have A Culture Of Cybersecurity*. https://www.forbes.com/sites/forbestechcouncil/2017/09/21/what-it-means-to-have-a-culture-of-cybersecurity/#6c5bd268efd1. (Accessed on 12/28/2017). Sept. 2017.

[50] C Hazay and Y Lindell. *Efficient oblivious polynomial evaluation and transfer with simulation-based security*. 2008.

[51] Carmit Hazay and Yehuda Lindell. "Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries." In: (2008), pp. 155–175.

[52] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[53] Carmit Hazay and Kobbi Nissim. "Efficient Set Operations in the Presence of Malicious Adversaries." In: *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*. 2010, pp. 312–331.

[54] Steven Hengel Jr., Sean Kern, and Andrea Little Limbago. "Operational Cyber Intelligence." In: *INSA, 2014* (2014).

[55] Martin Hirt, Ueli M. Maurer, and Bartosz Przydatek. "Efficient Secure Multi-party Computation." In: *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*. 2000, pp. 143–161.

[56] Gwoboa Horng. "An Efficient and Secure Protocol for Multi-party Key Establishment." In: *Comput. J.* 44.5 (2001), pp. 463–470.

[57] Yan Huang, David Evans, and Jonathan Katz. "Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?" In: *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. 2012.

[58] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. "Enhancing privacy and trust in electronic communities." In: *EC*. 1999, pp. 78–86.

[59] *ISAC's — NCSC*. https://www.ncsc.nl/english/Cooperation/isacs.html. (Accessed on 01/15/2018).

[60] Stanislaw Jarecki and Xiaomin Liu. "Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection." In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*. 2009, pp. 577–594.

[61] Stanislaw Jarecki and Xiaomin Liu. "Fast Secure Computation of Set Intersection." In: *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*. 2010, pp. 418–435.

[62] Tim van de Kamp, Andreas Peter, Maarten H. Everts, and Willem Jonker. "Private Sharing of IOCs and Sightings." In: *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, WISCS 2016, Vienna, Austria, October 24 - 28, 2016*. 2016, pp. 35–38.

[63] Tim Kelleher. *Locking Down the IT Security Framework: Predict, Prevent, Detect, Respond*. 2016.

[64] Simon Kemp. *The global state of the internet in April 2017*. https://thenextweb.com/contributors/2017/04/11/current-global-state-internet/. (Accessed on 12/26/2017). Apr. 2017.

[65] Pok-Son Kim and Arne Kutzner. "Ratio Based Stable In-Place Merging." In: *TAMC*. Vol. 4978. Lecture Notes in Computer Science. Springer, 2008, pp. 246–257.

[66] Ágnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. "Private Set Intersection for Unequal Set Sizes with Mobile Applications." In: *PoPETs* 2017.4 (2017), pp. 177–197.

[67] Lea Kissner and Dawn Xiaodong Song. "Privacy-Preserving Set Operations." In: *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*. 2005, pp. 241–257.

[68] Ronghua Li and Chuankun Wu. "An Unconditionally Secure Protocol for Multi-Party Set Intersection." In: *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*. 2007, pp. 226–236.

[69] Gaoqi Liang, Steven R Weller, Junhua Zhao, Fengji Luo, and Zhao Yang Dong. "The 2015 ukraine blackout: Implications for false data injection attacks." In: *IEEE Transactions on Power Systems* 32.4 (2017), pp. 3317–3318.

[70] Yehuda Lindell. "Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation." In: *J. Cryptology* 16.3 (2003), pp. 143–184.

[71] Yehuda Lindell and Benny Pinkas. "A Proof of Security of Yao's Protocol for Two-Party Computation." In: *J. Cryptology* 22.2 (2009), pp. 161–188.

[72] Helger Lipmaa. "Secure Accumulators from Euclidean Rings without Trusted Setup." In: *ACNS*. Vol. 7341. Lecture Notes in Computer Science. Springer, 2012, pp. 224–240.

[73] Nate Lord. *What are Indicators of Compromise?* https://digitalguardian.com/blog/what-are-indicators-compromise. (Accessed on 12/26/2017). July 2017.

[74] Lockheed Martin. "Cyber Kill Chain®." In: *URL: http://cyber.lockheedmartin. com/hubfs/Gaining_the_Advantage_Cyber_Kill _Chain. pdf* (2014).

[75] Rob McMillan. "Definition: threat intelligence." In: *Gartner, 2013* (2013).

[76] Ralph C. Merkle. "A Digital Signature Based on a Conventional Encryption Function." In: *CRYPTO*. Vol. 293. Lecture Notes in Computer Science. Springer, 1987, pp. 369–378.

[77] Rob van der Meulen. *Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016*. https://www.gartner.com/newsroom/id/3598917. (Accessed on 12/27/2017). Feb. 2017.

[78] James Alan Miller. *The top threat intelligence services for enterprises*. http://searchsecurity.techtarget.com/feature/The-top-threat-intelligence-services-for-enterprises. (Accessed on 01/15/2018).

[79] Steve Morgan. *2017 Cybercrime Report*. https://cybersecurityventures.com/2015-wp/wp-content/uploads/2017/10/2017-Cybercrime-Report.pdf. (Accessed on 12/27/2017). Oct. 2017.

[80] Steve Morgan. *Ransomware damages rise 15X in 2 years to hit £5 billion in 2017*. https://www.csoonline.com/article/3197582/leadership-management/ransomware-damages-rise-15x-in-2-years-to-hit-5-billion-in-2017.html. (Accessed on 12/27/2017). May 2017.

[81] Moni Naor and Benny Pinkas. "Efficient oblivious transfer protocols." In: *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.* 2001, pp. 448–457.

[82] Moni Naor and Omer Reingold. "Number-theoretic constructions of efficient pseudo-random functions." In: *Journal of the ACM (JACM)* 51.2 (2004), pp. 231–262.

[83] C. Andrew Neff. "A verifiable secret shuffle and its application to e-voting." In: *ACM Conference on Computer and Communications Security*. ACM, 2001, pp. 116–125.

[84] Bruce Nikkel and Stephan Glaus. *European Financial Institutes – Information Sharing and Analysis Centre, A Public-Private Partnership — ENISA*. https://www.enisa.europa.eu/topics/cross-cooperation-for-csirts/finance/european-fi-isac-a-public-private-partnership. (Accessed on 01/15/2018).

[85] Rasmus Pagh and Flemming Friche Rodler. "Cuckoo Hashing." In: (2001), pp. 121–133.

[86] Pascal Paillier et al. "Public-key cryptosystems based on composite degree residuosity classes." In: *Eurocrypt*. Vol. 99. Springer. 1999, pp. 223–238.

[87] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. "Information Theoretically Secure Multi Party Set Intersection Re-visited." In: *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*. 2009, pp. 71–91.

[88] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. "Round Efficient Unconditionally Secure MPC and Multiparty Set Intersection with Optimal Resilience." In: *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*. 2009, pp. 398–417.

[89] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing." In: *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. 1991, pp. 129–140.

[90] Kami Periman. *Ransomware Lessons for the Financial Services Industry*. https://blogs.cisco.com/financialservices/ransomware-lessons-for-the-financial-services-industry. (Accessed on 12/28/2017). May 2017.

[91] Benny Pinkas, Thomas Schneider, and Michael Zohner. "Faster Private Set Intersection Based on OT Extension." In: *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.* 2014, pp. 797–812.

[92] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. "Secure Two-Party Computation is Practical." In: vol. 2009. 2009, p. 314.

[93] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In: *Commun. ACM* 21.2 (1978), pp. 120–126.

[94] Kazue Sako and Joe Kilian. "Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth." In: *EUROCRYPT*. Vol. 921. Lecture Notes in Computer Science. Springer, 1995, pp. 393–403.

[95] Yingpeng Sang and Hong Shen. "Efficient and secure protocols for privacy-preserving set operations." In: *ACM Trans. Inf. Syst. Secur.* 13.1 (2009), 9:1–9:35.

[96] Martin De Saulles. *Internet of Things Definitions - What actually is the IoT?* http://informationmatters.net/internet-of-things-definitions/. (Accessed on 12/28/2017). Feb. 2017.

[97] Stephan van Schaik, Kaveh Razavi, Ben Gras, Herbert Bos, and Cristiano Giuffrida. "RevAnC: A Framework for Reverse Engineering Hardware Page Table Caches." In: *Proceedings of the 10th European Workshop on Systems Security, EUROSEC 2017, Belgrade, Serbia, April 23, 2017*. 2017, 3:1–3:6.

[98] Adi Shamir. "How to Share a Secret." In: *Commun. ACM* 22.11 (1979), pp. 612–613.

[99] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer, 2016.

[100] *Statistical new.indd*. https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2010.pdf. (Accessed on 12/27/2017). 2010.

[101] Michael Steiner, Gene Tsudik, and Michael Waidner. "Key Agreement in Dynamic Peer Groups." In: *IEEE Trans. Parallel Distrib. Syst.* 11.8 (2000), pp. 769–780.

[102] S. Joshua Swamidass and Pierre Baldi. "Mathematical Correction for Fingerprint Similarity Measures to Improve Chemical Retrieval." In: *Journal of Chemical Information and Modeling* 47.3 (2007), pp. 952–964.

[103] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N. Asokan. "The Circle Game: Scalable Private Membership Test Using Trusted Hardware." In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. 2017, pp. 31–44.

[104] Yuh-Min Tseng. "A Robust Multi-Party Key Agreement Protocol Resistant to Malicious Participants." In: *Comput. J.* 48.4 (2005), pp. 480–487.

[105] Jaideep Vaidya and Chris Clifton. "Secure set intersection cardinality with application to association rule mining." In: *Journal of Computer Security* 13.4 (2005), pp. 593–622.

[106] Abraham Waksman. "A Permutation Network." In: *J. ACM* 15.1 (1968), pp. 159–163.

[107] Andrew Chi-Chih 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.