Moving Multiparty Computation Forward for the Real World

Didem Demirag

A Thesis
in
The Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Information and Systems Engineering) at
Concordia University
Montréal, Québec, Canada

October 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Didem Demirag**

Entitled: **Moving Multiparty Computation Forward for the Real World**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Sébastien Le Beux, Ph.D.

_____ External Examiner
Abdelhakim Senhaji Hafid, Ph.D.

_____ External to Program
Emad Shihab, Ph.D.

_____ Examiner
Mohammad Mannan, Ph.D.

_____ Examiner
Amr Youssef, Ph.D.

_____ Thesis Supervisor
Jeremy Clark, Ph.D.

Approved by _____
Abdessamad Ben Hamza, Ph.D., Chair of Department

October 20, 2022 _____
Mourad Debbabi, Ph.D., Dean
Gina Cody School of Engineering and Computer Science

Abstract

Moving Multiparty Computation Forward for the Real World

Didem Demirag, Ph.D.

Concordia University, 2022

Privacy is important both for individuals and corporations. While individuals want to keep their personally identifiable information private, corporations want to protect the privacy of their proprietary data in order not to lose their competitive advantage. The academic literature has extensively analyzed privacy from a theoretical perspective. We use these theoretical results to address the need for privacy in real-world applications, for both individuals and corporations. We focus on different variations of a cryptographic primitive from the literature: secure Multi-Party Computation (MPC). MPC helps different parties compute a joint function on their private inputs, without disclosing them. In this dissertation, we look at real-world applications of MPC, and aim to protect the privacy of personal and/or proprietary data. Our main aim is to match theory to practical applications. The first work we present in this dissertation is a blockchain-based, generic MPC system that can be used in applications where personal and/or proprietary data is involved. Then we present a system that performs privacy-preserving link prediction between two graph databases using private set intersection cardinality (PSI-CA). The next use case we present again uses PSI-CA to perform contact tracing in order to track the spread of a virus in a population. The last use case is a genomic test realized by one time programs. Finally, this dissertation provides a comparison of the different MPC techniques and a detailed discussion about this comparison.

Acknowledgments

I would like to thank my advisor Dr. Jeremy Clark for his guidance and continuous support throughout my doctoral studies. I am very grateful that he always supported me to find the research topics that combined my various interests and provided me with the opportunities to improve myself academically. His motivation and positive attitude has always been a great source of encouragement for me. His passion for research and his dedication to share his extensive knowledge will continue to inspire me.

I would like to express my gratitude to my committee members Dr. Mohammad Mannan, Dr. Amr Youssef, Dr. Emad Shihab and Dr. Abdelhakim Senhaji Hafid for providing insightful suggestions and asking valuable questions that helped me improve my critical approach to research. I would also like to thank Dr. Erman Ayday for his continuous encouragement and support since my master's studies, and for his guidance throughout the different projects we worked on.

I would also like to thank my friends in the Madiba Security Research Group for the lively discussions we had about research, academia and many other topics. Montreal has a lively and welcoming language learning community and I would also like to thank all my friends who shared my passion for learning languages.

I would like to express my gratitude to my parents for their love, support and all the opportunities they presented to me. I would like to thank my dear grandmother, whose wisdom and kindness have a great influence on who I am now. Finally, I will always remember the loving memory of my grandfather who, with my grandmother, is the part of

my most heartwarming childhood memories.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main functions of cryptography are privacy (*e.g.,* encryption, key exchange), integrity (*e.g.,* MAC, signature) or both (*e.g.,* authenticated encryption, ZKP). In this dissertation, we focus on confidentiality, which is generally considered from the point of view of the users to protect their personally identifiable information. However, when we consider businesses, we realize that proprietary data also needs to be kept confidential, as it reveals critical information pertinent to the business itself. This need arises when data owners need to collaborate. For instance, data owners may prefer to perform a joint computation on their sensitive data with other data owners to better understand their own data. Some examples can be analysis done on healthcare, social network or banking data. While these data owners want to keep their clients' information confidential and keep their competitive edge, they also want to get insights from other data owners to offer better services to its users. This should be done in a way that none of the parties involved reveals their respective data. Keeping this need in mind, in this dissertation, we explore applications where personal and/or proprietary data needs to be kept confidential and how cryptography can help.

Secure multiparty computation (MPC) enables two or more parties to jointly perform a common computation on their respective private data without disclosing it to each other. As a very simple example, consider a setting where two professors at a university teaching two

different courses. They want to find the student with the highest grade across both courses in order to offer a scholarship. We can represent this setting as: $z = f(x, y)$, where f is the function to find the maximum grade, x and y are professors' respective inputs (*i.e.,* list of each student's grade) and z is the output of this function (the function can for example return the ID of the student with the highest grade). Everyone knows that the professors want to find the maximum grade. In other words, the function f is public. The trivial solution requires the professors to share their data with each other. However, they want to keep their data private. Hence, no one can compute z, as no one knows both x and y. MPC lets them collaborate to compute z.

Since 1980s, starting from the work of Yao [113], MPC has been extensively studied in the literature. What are the real world applications of MPC? How can we choose a specific MPC scheme that fits a certain application? There can be different flavours of MPC based on different aspects like the number of parties involved, whether the data is encrypted or secret-shared, whether the computation is outsourced to a third party. While choosing a suitable MPC scheme, how much information leak can be tolerated in a certain application and the efficiency should be considered. While an application that involves very sensitive data (*e.g.,* genomic data) will be more likely to tolerate less efficiency with a higher level of security, other applications may be able to tolerate a certain amount of information leak (*e.g.,* leaking limited proprietary data about social network graphs that does not make the corporation lose its competitive advantage).

Having these questions in mind, we explore four different applications: (i) a general MPC framework that also utilizes blockchain to perform MPC between two or more parties; (ii) a private set intersection cardinality (PSI-CA) based, privacy-preserving link prediction scheme among two different graph databases; (iii) an application for privacy-preserving contact tracing that is based on PSI-CA to protect healthcare data of individuals and (iv) an application that explores two different ways of using one time programs with a use case

focused on genomic testing. As (i) is a general purpose MPC, it can either be used in a setting with personal or proprietary data; (ii) is about proprietary data that needs to be protected for both businesses (*e.g.,* two social networks); (iii) aims to keep users' personal data (*i.e.,* health status) confidential; and (iv) is a blend of two: users' genomic data must be kept private, as well as the genomic test itself, as it is proprietary data of the company that designs the test.

The main theme in this dissertation is to analyze the different MPC schemes that we use in the upcoming chapters and discuss whether one way of doing MPC is the best, or whether it is the case that there are multiple sensible protocols each with its setting-specific advantages. As a part of this discussion, we also aim to analyze whether there are different considerations for using an MPC protocol and/or architecture when the protocol is run by individuals as opposed to companies. We provide this discussion in Chapter 7.

1.1 Contributions and outline

The rest of the dissertation is organized as follows: In Chapter 2, we provide an overview of the primitives and protocols used in this dissertation.

In Chapter 3, we describe Absentia, a blockchain-based approach for MPC. The emphasis of Absentia is reducing the participants' work to a bare minimum, where they can effectively have the computation performed in their absence and they can trust the result. While we use an MPC protocol (Mix and Match) that can operate perfectly well without a blockchain, the blockchain does add value in at least three important ways: (1) the MPC protocol requires a secure bulletin board and blockchains are the most widely deployed data structure with bulletin board properties (immutability and non-equivocation under reasonable assumptions); (2) blockchains provide a built-in mechanism to financially compensate participants for the work they perform; and (3) a publicly verifiable MPC protocol can be checked by the blockchain network itself, absolving the users of having to verify that the

function was executed correctly. We benchmark Absentia on Ethereum. While it is too costly to be practical (a single gate costs thousands of dollars), it sets a research agenda for future improvements. We also alleviate the cost by composing it with Arbitrum, a layer 2 'roll-up' for Ethereum which reduces the costs by 94%. Even though Arbitrum provides a significant reduction in cost , it is still not practical enough for evaluating bigger gates.

In Chapter 4, we propose a privacy-preserving link prediction scheme. Consider two data holders, ABC and XYZ, with graph data (*e.g.,* social networks, e-commerce, telecommunication, and bio-informatics). ABC can see that node A is linked to node B, and XYZ can see node B is linked to node C. Node B is the common neighbour of A and C but neither network can discover this fact on their own. In this chapter, we provide an MPC that ABC and XYZ can run to discover the common neighbours in the union of their graph data, however neither party has to reveal their plaintext graph to the other. Based on private set intersection, we implement our solution, provide measurements, and quantify partial leaks of privacy. We also propose a heavyweight solution that leaks zero information based on additively homomorphic encryption.

In Chapter 5, we address the privacy concerns that arise while tracking the spread of a virus in a population. With the Covid-19 pandemic, tracking and controlling the spread of a virus became a crucial need for almost all countries, as doing this early would save millions of lives and help countries keep a stable economy. The easiest way to control the spread of a virus is to immediately inform the individuals who recently had close contact with the diagnosed patients. However, to achieve this, a centralized authority (*e.g.,* a health authority) needs detailed location information from both healthy individuals and diagnosed patients. Thus, such an approach, although beneficial to control the spread of a virus, results in serious privacy concerns, and hence privacy-preserving solutions are required to solve this problem. Previous works on this topic either (i) compromise privacy (especially privacy of diagnosed patients) to have better efficiency or (ii) provide unscalable solutions.

In this chapter, we propose a technique based on a specific type of MPC called private set intersection between physical contact histories of individuals (that are recorded using smart phones) and a centralized database (run by a health authority) that keeps the identities of the positively diagnosed patients for the disease. The proposed solution protects the location privacy of both healthy individuals and diagnosed patients and it guarantees that the identities of the diagnosed patients remain hidden from other individuals. Notably, proposed scheme allows individuals to receive warning messages indicating their previous contacts with a positively diagnosed patient. Such warning messages will help them realize the risk and isolate themselves from other people. We make sure that the warning messages are only observed by the corresponding individuals and not by the health authority. We also implement the proposed scheme and show its efficiency and scalability via simulations.

In Chapter 6, we present a genomic testing application that utilizes one time programs (OTP). An OTP works as follows: Alice provides Bob with the implementation of some function. Bob can have the function evaluated exclusively on a single input of his choosing. Once executed, the program will fail to evaluate on any other input. State-of-the-art one-time programs have remained theoretical, requiring custom hardware that is cost-ineffective/unavailable, or confined to adhoc/unrealistic assumptions. To bridge this gap, we explore how the Trusted Execution Environment (TEE) of modern CPUs can realize the OTP functionality. Once this functionality is realized, OTP can be used in different use cases. In this chapter, we show how genomic testing can be performed using OTP. We also provide an overview of other use cases that utilize OTP. We present two different implementations of OTP: in the first implementation, the TEE directly enforces the one-timeness of the program and the genomic test is run only once in TEE; and in the second one, the instantiation of OTP is based on a cryptographic primitive called one-time memory and the program is represented with a garbled circuit. These options have different performance profiles: the first is best when Alice's input is small and Bob's is large, and the second

for the converse. In our genomic use case, Alice (the vendor) provides Bob with a device that has the genomic test. Bob can then provide his input (his sequenced genome) to the device to learn the result of the test. Our evaluations show that the first variant has a better performance for the genomic use case, as Alice's input is small and Bob's is large. Finally, we provide a discussion about the suitable OTP variant for each proposed use case.

In Chapter 7, we summarize the main takeaways from each chapter. We present a comparison of different MPC techniques used in these applications. Finally, we discuss possible future directions.

1.2 Other publications

In this section, we present other projects that we do not cover in this dissertation:

- Mapping the Privacy Landscape for Central Bank Digital Currencies: Now is the time to shape what future payment flows will reveal about you [7]: Can a central bank digital currency provide complete privacy to its users while remaining auditable in the event of abuse? Minimising the conflicts of interest between privacy-conscious users, data holders, and law enforcement requires to differentiate between soft (*i.e.,* institution-based) and hard (technology-based without discretion) solutions for privacy and auditability. From this vantage point, we investigate the various proposed privacy-enhancing technologies and the underlying trade-offs. We demonstrate that the promise of buzzword concepts like zero knowledge proofs is, on the whole, elusive. To date, all solutions with rule-based auditability make it trivial for criminals to change their behaviour to avoid prosecution. Real-life crime fighting uses a plethora of methods, each of which conflicts with privacy goals to some extent. Unless better technological solutions emerge, a realistic approach is one grounded in strong institutions and supported with technically enforced access control. One example

is the public sector offering a basic layer of soft privacy, protected by a good legal envelope, limited retention periods, and audits.

- Opening Sentences in Academic Writing: How Security Researchers Defeat the Blinking Cursor [47]: Traditionally, education in computer science focuses on stakeholders like teachers, undergraduate students, and employers. However researchers also educate themselves about recent results and new subject matters. An important vehicle in this informal, self-education process is reading peer-reviewed academic papers—papers that are also used in the curriculum of graduate-level research courses. Technical writing skills are important in this domain, as well as engaging the reader with interesting text. This paper is a study of academic writing. We study in depth the first sentence used by researchers in opening their academic papers and how this sentence operates to draw the reader in. We use a corpus of 379 papers from a top-tier cybersecurity conference and use qualitative analysis (coding from grounded theory) to create a taxonomy of 5 general types and 14 sub-types of opening sentences. In this paper, we define and illustrate each type through examples, and reflect on what we learned about writing after examining all of these sentences.

- Demystifying Stablecoins: Cryptography meets monetary policy [34] (For the full paper, we refer the reader to [35]): This papers aims to provide an understandable survey of 'stablecoins'—cryptocurrencies designed to have lower volatility than Bitcoin or Ether. We distil the stability mechanisms into a set of primitives and precisely analyze why they are thought to adjust exchange rates, what assumptions they are based on, and what risks still exist. Stablecoins are interesting to the cryptocurrency community because volatility has hampered spending, increased speculation, and hindered mature lending/credit markets from forming. Our comparison includes pre-Bitcoin digital currencies like Liberty Reserve and e-gold, Ethereum's gas, and novel visualizations.

Chapter 2

Background

In this chapter, we explain the basic concepts and cryptographic primitives we use in this thesis.

2.1 Blockchain

Pseudonymous Satoshi Nakamoto introduced, in his white paper [88], blockchain as the underlying technology of the cryptocurrency Bitcoin. Blockchain is a tamper-proof ledger with non-equivocation and immutability properties. Each user participating in the network can post transactions. The validity of transactions are checked by every node before it is permanently recorded on the blockchain inside a new 'block'. The whole network reaches a 'consensus' that a certain transaction can be added to the chain. Without controlling the majority of the computing power of the network, the records cannot be changed retrospectively (*i.e.,* the ledger is tamper-proof). The ledger is also publicly-verifiable—everyone in the network can see and verify the records themselves. Each member of the peer-to-peer network has a copy of the ledger which is not governed by any central authority. Both blockchain and cloud computing provide a way to offload computation. While blockchain has high integrity and correctness guarantees, it has low efficiency as every node has to

perform the same computation. Cloud computing presents a more efficient solution but unlike blockchain, the trust is centralized.

Bitcoin. The decentralized digital currency Bitcoin aims to solve the pitfalls of the centralized payment systems like banks or other financial institutions. There is no central authority that issues Bitcoin or checks the transactions. The transactions are kept permanently in the tamper-proof Bitcoin blockchain. Each transaction that is created by a participant is first verified by other participants of the network using proof of work mechanism, which involves the computation of a 'mathematical puzzle' whose difficulty is adjusted by the system to rate-limit the creation of new blocks which records transactions. Solving this puzzle to extend the chain is called *mining*. A new block is introduced to the system approximately in every 10 minutes. Proof of work also provides fault tolerance for the network [78].

Ethereum. The limited capabilities of the scripting language of Bitcoin has led to the creation of Ethereum blockchain that enables a user to program with a Turing-complete language. Ethereum is created by Vitalik Buterin in 2014 [21]. Compared to Bitcoin, block creation time is much faster, 10-20 seconds, in Ethereum. In addition to being a system for the cryptocurrency Ether, Ethereum blockchain also allows users to write programs in Solidity, which is a high level programming language, and deploy these applications in the decentralized, peer-to-peer network. These applications are named as *decentralized applications* (DApps). DApp in a blockchain runs through smart contracts. Smart contracts are written in Solidity and they are deployed with the help of Ethereum Virtual Machine (EVM). The code cannot be changed after a smart contract is deployed. All nodes in the network execute the smart contract and they all reach the same state. This is needed to ensure that the contract is executed correctly. The changes in the state are atomic. Anyone can re-perform the computation to verify that the computation is done correctly—the contracts are publicly verifiable. The functions in a smart contract cannot run on their own,

they should be explicitly called. Transactions in Ethereum costs a certain amount of fee measured in *gas*. The price of gas is determined based on an auction mechanism. Miners are more likely to accept to execute a transaction if a higher gas value is bid for it.

2.2 Common adversarial assumptions

In this section, we explain the common adversarial assumptions that we refer to in the upcoming chapters.

- Semi-honest: A semi-honest adversary follows the protocol honestly, but can be curious to learn sensitive information pertaining to other participants in the protocol. This assumption is used in Chapter 4, Chapter 5, and Chapter 6.
- Malicious: A malicious adversary can arbitrarily deviate from the protocol. This assumption is used in Chapter 3.
- Public verifiability: If a participant in the protocol exhibits adversarial behaviour, it can be detected not only by the participants of the protocol but by anyone, in a publicly verifiable system (also called publicly auditable or universally verifiable). This assumption is used in Chapter 3.

While for some of the real-world applications semi-honest setting is enough as a requirement, for others this may not be the case. However, semi-honest model is a stepping stone for the malicious model. There can be techniques (*e.g.,* adding zero knowledge proof at every step) that takes a semi-honest setting and makes it malicious.

2.3 Encryption

Encryption allows a sender to keep the message hidden from other parties during its transmission to the receiver, who can retrieve the message with the private key. In this section,

first the mathematical setting, then Elgamal encryption scheme is explained [53]. The setting that we use is the multiplicative subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p^*$, where p and q are large primes. Note that the computations that are shown in the rest of the chapter are all $mod\ p$, unless otherwise is specified.

Mathematical setting.

- **Discrete logarithm problem (DLP):** Let $g \epsilon \mathbb{G}_q$ be a generator and $a, b, z \epsilon \mathbb{Z}_q$ are randomly chosen exponents. DLP is to compute a given $\langle g, g^a \rangle$.
- **Decisional Diffie Hellman (DDH) problem:** is to distinguish $\langle g, g^a, g^b, g^{ab} \rangle$ from $\langle g, g^a, g^b, g^z \rangle$.
- **Elliptic Curves:** An elliptic curve E over F_p is the set of points in the form (x, y) that satisfy the equation $E = \{(x, y) \epsilon \mathbb{Z}_p \times \mathbb{Z}_p \mid y^2 \equiv x^3 + ax + b\ (mod\ p)\} \cup \{\mathcal{O}\}$ where $a, b \epsilon \mathbb{Z}_p$, $4a^3 + 27b^2 \neq 0\ (mod\ p)$, $p > 3$ and $\mathcal{O}$ is the point at infinity. Absentia, which is described in Chapter 3, uses Elgamal over elliptic curve `secp256k1` where DLP and DDH-problem are hard. Note that, we use the conventional multiplicative notation $y = g^x$ instead of $Y = xG$.

Elgamal cryptosystem. Elgamal is a public-key cryptosystem [54]. The cryptosystem is defined over the multiplicative subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p^*$. $\mathbb{G}_q$ has order q. x, $y = g^x$ are the private and public keys respectively, where x is chosen uniformly random from $\mathbb{Z}_q$ and g is the generator of $\mathbb{G}_q$. For a message $m \epsilon \mathbb{G}_q$, encryption with the public key works as $Enc(m) = (c_1, c_2) = (g^a, my^a)$ where a is chosen uniformly random from $\mathbb{Z}_q$. Decryption with the private key is defined as $Dec(c_1, c_2) = (c_1)^{-x} c_2 = m$.

Exponential Elgamal. Exponential variant of Elgamal encrypts g^m instead of m to obtain $(c_1, c_2) = (g^a, g^m y^a)$. Exponential Elgamal is additively homomorphic, meaning that

two encrypted ciphertexts can be multiplied together to obtain the sum without decrypting them: $Enc(m_1) * Enc(m_2) = Enc(m_1 + m_2)$. Note that, decryption requires solving a discrete logarithm, so plaintexts must stay small (*e.g.,* under a trillion) .

Rerandomization. Rerandomization function takes an existing ciphertext and converts it to another ciphertext that represents the encryption of the same message with a different randomness. Rerandomization does not require the knowledge of the private key, the previous randomness, or the message that is encrypted. The ciphertext $(c_1, c_2) = (g^a, my^a)$ is rerandomized as $(c_1', c_2') = (g^{a'}, my^{a'})$. $\langle g^a, my^a, g^{a'}, my^{a'} \rangle$ is a DDH-tuple if the message m is not changed. Under the DDH-problem, rerandomized ciphertext cannot be distinguished from an encryption of another message without the knowledge of the private key x or the random value a'.

Distributed decryption. In distributed decryption the private key is shared among n parties. Distributed key generation (DKG) generates a public key and shares of the private key for each participant.

In the distributed version, n out of n parties are needed to perform the decryption for any integer n (*e.g.,* 5 out of 5). n parties involved in the protocol own a share of the private-public key pair from the set $\{(x_1, y_1), (x_2, y_2) \ ... \ (x_n, y_n)\}$. The public key is defined as $\prod_{i=1}^{n} y_i$ and the private key is defined as $\sum_{i=1}^{n} x_i$. Each party partially decrypts the ciphertext with their respective share of the private key: $c_2 c_1^{-x_i}$.

The parties also prove that they performed the correct exponentiation with their share of the private key. To do so, they use zero knowledge proof. Full decryption can be done if all parties perform decryption with their private keys $c_2(c_1^{-x_1} * c_1^{-x_2} * ... c_1^{-x_n}) = c_2 c_1^{-\sum_{i=1}^{n} x_i}$.

Distributed decryption can be transformed into threshold decryption [91, 49, 58] where $m < n$ participants are needed to recover the private key and decrypt the ciphertext.

2.4 Σ-protocols

Σ-protocol defines an interactive and three move proof of statement between the prover and the verifier. At the end of the protocol, the verifier only learns whether the statement is correct [37, 63]. They are a subtype of zero knowledge proofs.

Schnorr. Given $\mathbb{G}_q$, the generator $g \epsilon \mathbb{G}_q$ and group element $y \epsilon \mathbb{G}_q$, the prover proves the knowledge of w, discrete logarithm $log_g y$, in $y = g^w (mod\ p)$. Public parameters (pp) are defined as $pp = \langle g, q, p \rangle$. The knowledge of w, the discrete logarithm, is described by Schnorr [98]. Prover randomly chooses $a \epsilon \mathbb{Z}_q$ and computes $b = g^a$ and sends to the verifier. Verifier chooses a random challenge $c \epsilon \mathbb{Z}_q$ and sends to the prover, who then computes $d = cw + a\ (mod\ q)$ and sends to the verifier. $\langle b, c, d \rangle$ is called the transcript and the verifier accepts the transcript if and only if $g^d = by^c$.

Fiat-Shamir. In Fiat-Shamir, the prover runs the Σ protocol himself and anyone can verify the soundness of the accepting transcript [55]. This is achieved by setting $c = H(b, y, pp)$ which makes sure that the value b cannot be chosen after c is chosen. Therefore, the proof cannot be replayed in a different setting, as the public parameters are locked into the proof.

Non-interactive Chaum-Pedersen. Prover aims to prove the knowledge of w such that $y_1 = g_1^w$ and $y_2 = g_2^w$ where g_1, g_2 are generators, y_1, y_2 are group elements and statement is defined as $stmt = (g_1, g_2, y_1, y_2)$ [27]. Prover chooses $a \epsilon \mathbb{Z}_q$; computes $b_1 = g_1^a$, $b_2 = g_2^a$ and sends these values to the verifier. Verifier chooses the challenge $c \epsilon \mathbb{Z}_q$ and sends to the verifier, in the interactive version. Prover computes $d = cw + a\ (mod\ q)$. The proof is accepted by the verifier if $g_1^d = {y_1}^c b_1$ and $g_2^d = {y_2}^c b_2$.

In the non-interactive version of the protocol, the challenge is computed by the prover as $c = H(b_1, b_2, stmt, pp)$, where H is the cryptographic hash function. As an Elgamal

ciphertext and its rerandomization forms a DDH-tuple, Chaum-Pedersen can be used as a proof for correct rerandomization.

2.5 Secure multiparty computation

In 1982, Yao explained a solution for secure two-party computation, which is known as *Yao's Millionaire problem* [113]. In this setting, there are two parties, that want to decide who is wealthier. Directly disclosing their respective wealths would be the most trivial solution, but not privacy-preserving. *Yao's Millionaire problem* specifies the setting where two parties, Alice and Bob, aim to find out who is wealthier without disclosing their respective wealth. This is characterized as secure two-party computation in the literature. At the end of an interactive protocol involving several rounds of exchange of data, they can find out who is wealthier. The generalization of this problem to n-parties is known as secure multi-party computation (MPC). All parties should be online to perform MPC and if a party goes offline, the computation stops until that party is online again.

While there are many different approaches to realize MPC, we focus on focus on (i) garbled circuits, (ii) Mix and Match, (iii) private set intersection cardinality (PSI-CA), and (iv) authorized private set intersection (APSI). In Chapter 7, we provide a comparison of the following MPC techniques.

2.5.1 Garbled circuits

Garbled circuits are introduced by Yao [113]. The function to be evaluated by two (or more) parties should first be represented as a boolean circuit. To illustrate, we use a NAND gate:

A	B	Out
0	0	1
1	0	1
0	1	1
1	1	0

Before we explain how the function is evaluated, we first introduce **oblivious transfer (OT)**, which is a protocol that provides one way for Alice and Bob to perform MPC. For OT, Alice has a list of inputs as $(x_1, ..., x_n)$ and Bob has chosen one input $i \epsilon 1, ..., n$ to receive. OT allows Bob to receive x_i from Alice's list, while Alice remains oblivious to i and Bob does not learn anything about any other x in Alice's list.

To build MPC using OT, Alice first encrypts the rows of the table, to do that she generates a key for each wire j: k_j^0 and k_j^1, where $j \epsilon \{a, b\}$. Symmetric encryption is used in this setting.

A	B	Out
k_a^0	k_b^0	$Enc_{k_a^0}(Enc_{k_b^0}(k_c^0))$
k_a^1	k_b^0	$Enc_{k_a^1}(Enc_{k_b^0}(k_c^1))$
k_a^0	k_b^1	$Enc_{k_a^0}(Enc_{k_b^1}(k_c^1))$
k_a^1	k_b^1	$Enc_{k_a^1}(Enc_{k_b^1}(k_c^0))$

Bob receives the garbled gate along with the encrypted inputs from Alice (let's assume the values are k_a^0 and k_b^0). Bob directly receives k_a^0 from Alice. Alice and Bob use oblivious transfer for Bob to receive k_b^0. Alice remains oblivious to the fact that Bob received k_b^0. The keys received by Bob allow him to reveal only one row as the result. Garbled circuit is the basis of one-time program (OTP) which will be introduced in Chapter 6.

2.5.2 Mix and match

Having looked at garbled circuits, which uses symmetric key encryption, in this section we explain Mix and Match, which is akin to a public key version of garbled circuits. Using public keys makes it easier for the participants to prove in zero knowledge that they are following the protocol correctly. Here, we use threshold decryption with shared keys. Garbled circuit is faster than a Mix and Match protocol, as it uses symmetric key encryption. Mix and Match addresses the following shortcomings of garbled circuits: (i) while in Mix and Match, there is a proof that the circuit is constructed correctly, for garbled circuits the person who constructed the circuit is trusted; (ii) oblivious transfer is not secure against malicious adversaries and as a result, neither is a garbled circuit; (iii) in order to evaluate a garbled circuit, both Alice and Bob has to be online. However, in Mix and Match the steps can be handled by chosen parties (we call them trustees) on behalf of Alice and Bob. This is possible, because it is publicly verifiable.

Even though Mix and Match is way more expensive, compared to garbled circuits, it provides security in the malicious setting, as every step is covered with zero knowledge proofs to prove the correctness of the computation. While Mix and Match has some advantages over garbled circuits, we cannot definitively say that it is a replacement for garbled circuits. On the other hand, garbled circuit is a fast solution in a two-party, semi-honest setting.

In this section, we give an overview of Mix and Match protocol by Jakobsson and Juels [68] which provides a generic construction for MPC. Mix and Match uses a partially homomorphic encryption scheme. In Chapter 3, we explain how we instantiate it with additive exponential Elgamal [38] over elliptic curves (specifically `secp256k1` which is used natively by Ethereum).

Mix and Match: Pre-computation. In a pre-computation stage, the following tasks are completed. First, a set of n trustees, identified by public keys, are chosen. A threshold of trustees needed to complete the protocol can also be chosen, however we implement the simplest case: 2-out-of-2 (we call this *distributed* as opposed to *threshold*). Next, the trustees use a distributed key generation (DKG) protocol for creating n shares of the decryption key, one for each trustee, as well as a single joint public key. Exponential Elgamal supports DKG and threshold decryption [91].

In Mix and Match, a circuit of the function to be evaluated is produced using multi-input and multi-output lookup tables. We evaluate a single binary NAND gate (a universal gate that can create any circuit) which corresponds to a lookup table with two binary inputs (one from Alice and one from Bob) and a single binary output. During a pre-computation stage, the circuit for the function is established as a sequence of lookup tables (the output from one table can be used as an input to another). Each element of each lookup table is individually encrypted under the trustees' public key (we denote an encryption of x as $[\![x]\!]$):

A	B	Out
$[\![0]\!]$	$[\![0]\!]$	$[\![1]\!]$
$[\![1]\!]$	$[\![0]\!]$	$[\![1]\!]$
$[\![0]\!]$	$[\![1]\!]$	$[\![1]\!]$
$[\![1]\!]$	$[\![1]\!]$	$[\![0]\!]$

The encrypted table is then permuted row-wise. Each trustee mixes the rows, rerandomizes each ciphertext, and proves in zero knowledge that the result is correct:

A	B	Out
$[\![0]\!]$	$[\![1]\!]$	$[\![1]\!]$
$[\![1]\!]$	$[\![0]\!]$	$[\![1]\!]$
$[\![1]\!]$	$[\![1]\!]$	$[\![0]\!]$
$[\![0]\!]$	$[\![0]\!]$	$[\![1]\!]$

Complete circuits of such tables can be pre-computed by the trustees before Alice and Bob provide their inputs. Practically speaking, if sets of trustees were pre-established, they could prepare circuits for commonly requested functions and post them publicly. When Alice and Bob decide to do an MPC, they can choose the pre-computed circuit (produced by a specific set of trustees).

Plaintext Equality Test (PET). Let $\langle [\![x]\!], [\![y]\!] \rangle$ denote two exponential Elgamal ciphertexts of x and y respectively. The trustees first compute $[\![z]\!] = [\![x - y]\!]$ using the additively homomorphic property. If the values are the same, $z = 0$; otherwise $z \neq 0$. Each trustee chooses a random $r_i \neq 0$, where $r_i \epsilon \mathbb{Z}_q$, computes $[\![\hat{z}_i]\!] = [\![r_i * \hat{z}_{i-1}]\!]$ (where $\hat{z}_0 := z$) and proves correctness in zero knowledge. The resultant $[\![\hat{z}]\!] = [\![\prod r_i * z]\!]$ will still be $[\![0]\!]$ when $x = y$ and will encrypt a randomly distributed non-zero integer otherwise. (The original proposal [68] lets each trustee blind without using the result from the previous trustee, which adds asynchronicity but requires a critical security correction [84]). In the final step, the trustees decrypt and reveal $\hat{z}$. If $\hat{z} = 0$, the equality test returns `True`; and returns `False` otherwise.

Mix and Match: Online phase. At online phase, Alice and Bob provide their inputs $\langle [\![a]\!], [\![b]\!] \rangle$. The trustees compute a PET between $[\![a]\!]$ (Alice's input) and each ciphertext in the column corresponding to Alice's input. They do the same for Bob. They locate the row that returns true for every input column. There can be three options for the encrypted output(s) of this row: it can be (1) transferred as an input to the next gate, (2) decrypted publicly if it is a final output, or (3) proxy re-encrypted for Alice (and/or Bob) which obliviously and verifiably changes the output by the trustees from an encryption under the trustees' joint public key to an encryption under Alice's. We illustrate for the previous example and $a = 1$ and $b = 0$:

A	B	Out
$\mathsf{PET}(\llbracket a \rrbracket, \llbracket 0 \rrbracket) = \mathtt{F}$	$\mathsf{PET}(\llbracket b \rrbracket, \llbracket 1 \rrbracket) = \mathtt{F}$	$\llbracket 1 \rrbracket$
$\mathsf{PET}(\llbracket a \rrbracket, \llbracket 1 \rrbracket) = \mathtt{T}$	$\mathsf{PET}(\llbracket b \rrbracket, \llbracket 0 \rrbracket) = \mathtt{T}$	$\llbracket 1 \rrbracket$ is selected
$\mathsf{PET}(\llbracket a \rrbracket, \llbracket 1 \rrbracket) = \mathtt{T}$	$\mathsf{PET}(\llbracket b \rrbracket, \llbracket 1 \rrbracket) = \mathtt{F}$	$\llbracket 0 \rrbracket$
$\mathsf{PET}(\llbracket a \rrbracket, \llbracket 0 \rrbracket) = \mathtt{F}$	$\mathsf{PET}(\llbracket b \rrbracket, \llbracket 0 \rrbracket) = \mathtt{T}$	$\llbracket 1 \rrbracket$

2.5.3 Private set intersection cardinality

Private set intersection cardinality (PSI-CA) aims to compute the cardinality of the intersection of two sets belonging to two parties, namely client and server, without disclosing either set to the other party [42]. At the end of the protocol between the client and server, the client learns only the size of the intersection. The only information that is leaked to the respective parties is the upper bounds for the size of the client's and server's inputs. We provide the details of the PSI-CA algorithm in Chapter 4 and Chapter 5 in the context of the proposed solutions.

The previous two primitives work for any type of circuit and hence, it can also work for PSI-CA. However, if the setting is known ahead of time (two party and semi-honest), PSI-CA is preferred over these options. It is more efficient, as it is created for a specific purpose.

Mathematical setting for PSI-CA.

- PSI-CA is described in the multiplicative subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p^*$ where p and q are large primes and $q \mid p-1$.
- The hash functions used in PSI-CA is defined as $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, where κ is the security parameter. The hash functions are modeled as random oracles.

2.5.4 Authorized private set intersection (APSI)

Similar to PSI-CA, authorized private set intersection (APSI) also aims to compute the intersection of two sets (belonging to two parties) in a privacy-preserving way [43]. Different from PSI-CA, APSI requires the authorization of the client input first. Thus, the client input is signed by a mutually trusted authority and the client also provides these signatures as the input of the algorithm. We provide the details of the APSI algorithm in Chapter 5 in the context of the proposed solution.

Mathematical setting for APSI.

- APSI is described in the setting with RSA modulus $N = pq$ where p and q are safe primes ($p = 2q + 1$), public exponent e, a random element g in $\mathbb{Z}_N^*$. The hash functions H and H' are modeled as random oracles. The computations in APSI protocol are $mod\ N$.
- The hash functions used in APSI is defined as $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, where κ is the security parameter.

RSA. Let RSA modulus be $N = pq$ where p and q are prime numbers. The security of RSA is based on the difficulty of factoring the product of two large primes. $\phi(N) = (p-1)(q-1)$, where $\phi(N)$ is Euler's totient function. Public exponent e is chosen such that $1 < e < \phi(N)$ and gcd(e, $\phi(N)$) $= 1$. Private key exponent d is defined as $d = e^{-1}\ (mod\ \phi(N))$. N and e are the public values. Encryption of message m is defined as $c = m^e\ (mod\ N)$ and decryption is done as follows: $m = c^d\ (mod\ N)$.

RSA Signature. Key generation is the same as the RSA encryption scheme. Signature over message m is $\sigma = Sign_d(m) = m^d\ (mod\ N)$. The verification is defined as $Verify_e(m, \sigma) = \sigma^e\ (mod\ N) \stackrel{?}{=} m$

Chapter 3

Secure multiparty computation using blockchain

This chapter is based on the work supervised by Jeremy Clark and it is published at 2021 Workshop on Trusted Smart Contracts (WTSC) co-located with Financial Cryptography and Data Security (FC) [46] .

3.1 Introduction

Consider the traditional setting for multiparty computation (MPC) with a twist: Alice and Bob each have some data, they would like to know the output from running an agreed-upon function on their data, each does not want the other (or anyone else) to learn their data, *and* they want to simply submit their data (*e.g.,* encrypted) to a trustworthy system and come back later for the result, which will always be correct. They are willing to pay for this service and they accept that, only in the worst case of full collusion between the operators of this service (called trustees), their inputs may be exposed—but a single honest trustee protects their privacy.

Recall blockchain technology, Ethereum, and smart contracts or decentralized apps

(DApps) from Chapter 2. Can these technologies help? In theory? In practice? We seek to answer these questions through direct experimentation. Blockchain can help in three different aspects: (1) it provides an integral point of coordination where trustees can post and track progress on the evaluation; (2) it provides an in-band solution for paying the trustees (in either a cryptocurrency like ETH or in a stablecoin pegged to the value of governmental currency like the USD) in a way that is contingent on their performance; and (3) the blockchain itself can serve as the public verifier and can reject any protocol proof that is not correct. When Alice and Bob retrieve the result (whether in plaintext or individually encrypted under their keys), they know it must be correct—otherwise it would not be there waiting for them.

Our experiments show that while in theory the idea is sound and we are able to successfully perform a secure function evaluation of a single logic gate (NAND gate) on Ethereum, the costs today are too prohibitive for it to be considered practical. We then turn to so-called layer-2 solutions and show that Arbitrum [71] can make Absentia substantially more practical (with room for further improvement). Even though Arbitrum reduces Ethereum gas costs by 94%, it is still not practical enough for evaluating bigger gates.

3.1.1 Key design decisions

Design decision: Trustee model. In keeping with our priority for a submit-and-go protocol, someone has to perform the actual evaluation of the function on the inputs. We call these entities *trustees*. We require that the number of trustees (n) can be chosen independently of the number of inputs. In Absentia, we assume all trustees (n-out-of-n) participate (and can identify any that do not). However Absentia could be modified to allow the protocol to proceed if only a threshold (t out of n) of trustees participate—however, also reduces the number of trustees that need to collude to break the privacy of the protocol.

The remaining question is how can Alice and Bob find trustees they assume will not

collude? We have several suggestions: (1) it could be based on personal connections; (2) perhaps commercial entities would emerge with either pre-established reputations or earn their reputation over time (similar to oracle providers today); confidence might increase if they offer legally enforceable terms of service; or (3) trustees could be picked at random from a large set of trustees. While (3) may not sound convincing, it is essentially same threat model as the anonymous web-browsing tool Tor which is trusted by many vulnerable users. For option (1), a user has to personally have trusted connection that possess the required technical setting to perform the computation. These personal connections should be trusted by both parties that provide their private inputs for computation. Overall, this option is less usable compared to others. Option (2) depends on whether there is an incentive for a business to provide a service like this. Finally, option (3) provides the most usable solution, as users do not have to find the trustees themselves. However, this option has some disadvantages in terms of trust. It becomes more likely that there will be a collusion between the trustees. As mentioned before, this model works for Tor. As determining the trustees is not one of the main contributions of this chapter, we do not provide a more detailed discussion of this aspect.

Design decision: Ethereum. While we are not the first to explore multi-party computation and its relationship to blockchain (see Section 3.2.1), we believe we are the first to implement an MPC protocol on a public, commonly used blockchain; namely, Ethereum. The first research question we ask is whether MPC is even feasible on Ethereum, given the heavy cryptography it uses. Our proposed solution establishes a benchmark that we hope to see improved through future research. Ethereum itself has scheduled scalability plans including Ethereum 2.0 (more transactions per second), and a lot of community resources are also being spent examining and implementing *layer 2* solutions that move blockchain functionality off of the main chain without sacrificing many of its security benefits. Technologies include *state channels*, *sidechains*, and *roll-ups* [61]. To experiment

with these technologies, we also deploy and benchmark critical components of Absentia on Arbitrum [71], a recently proposed system for optimistic roll-ups (described more in Section 3.4). We now turn to another avenue for improvement, using state-of-the-art MPC protocols.

As the MPC protocol requires a secure bulletin board, we use a blockchain for this purpose. Bulletin boards were proposed before blockchains [36, 92, 65] but these were never actually deployed. Blockchains have all the security properties of a bulletin board (plus additional properties that allow for computing smart contracts, not just recording data). Specifically, blockchains provide immutability (data cannot be changed once written) and non-equivocation (once data is confirmed, all parties see the same data) under reasonable assumptions. Another reason we prefer Ethereum is that the participants can easily be financially compensated, as Ethereum provides a built-in mechanism for this purpose. However, with another bulletin board payments have to be implemented as a separate system as opposed to Absentia, where the computation and the payment are atomic.

Design decision: Mix and match. Starting with Yao in 1982 [113], the question of how to securely evaluate a general function, when inputs are held by multiple people, has generated a rich body of literature in cryptography. In choosing an MPC protocol for the basis of Absentia, we looked for one with the following properties:

1. **Trustee model.** As justified above, we seek an MPC protocol that lets the input holders (*e.g.,* Alice and Bob) offload their inputs to a set of non-colluding trustees for evaluation.

2. **Publicly verifiable** (*a.k.a.* publicly auditable or universally verifiable). Many MPC protocols are in the semi-honest (*i.e.,* honest-but-curious) model. Some are resilient to covert or malicious adversaries. We require that not only can adversarial behaviour be detected by the participates in the protocol, but that it can be detected by anyone

(*i.e.,* the public). This allows (a) Alice and Bob to offload the computational work to the trustees and still ensure the output is correct, even if they did not directly participate, and (b) Alice and Bob can go further and offload the verification itself to someone they trust—the Ethereum network in this case.

3. **Identifiable aborts.** If the protocol does not reach completion, anyone can establish which trustee aborted. Financial incentives can be attached to participation and timeliness.

4. **Elliptic curve operations.** While Ethereum can in theory implement different types of cryptography (RSA groups, integer-based discrete logarithms groups, lattices, *etc.*), it has native support for its own cryptographic operations (ECDSA signatures) on the elliptic curve `secp256k1`. For ease of implementation, we prefer a MPC with the same cryptographic setting.

5. **Circuit type.** When the function to be evaluated is represented as a circuit, the circuit could be based on logic gates (*i.e.,* NAND gates) or arithmetic operations (*e.g.,* additions and multiplications in a modular group). We are indifferent to this design parameter.

One MPC protocol to meet our purposes is Mix and Match [68] and we chose it based on our familiarity with it. We are also aware that the state-of-the-art MPC protocols are based on a different paradigm—based on *Beaver triples* [12]—initiated by the SPDZ protocol [41, 40] with many followups (HighGear is a recent example [74]). While SPDZ uses lattice-based somewhat homomorphic encryption (SME), this is during a pre-computation phase and Absentia (for now) assumes all pre-computation has been validated. SPDZ also appears amenable to a trustee model and one paper explores a publicly verifiable variant [10], however since the authors do not compare themselves to Mix and Match, it would

be a full research project to determine if it is indeed faster. We note that it is not obviously categorically faster—for example, by not requiring public key operations at all: the publicly verifiable variant uses Pedersen commitments extensively.

We are not aware of an explicit *proof* that Mix and Match is publicly verifiable, however every step of the protocol is covered by a trustee issuing a non-interactive zero knowledge proof and it is later assumed to be by the authors in their auction application [68]. Stated a different way, it appears that even when all trustees fully collude, trustees can only break privacy (and not integrity) with the exception of one sub-protocol, as noted by the authors [68], called the *plaintext equality test* (PET). Despite the caveat, many have used the PET protocol as if it is publicly verifiable (some making justifications based on statistical arguments). Recently it was shown these statistical arguments are not sufficient, but the PET protocol can be made verifiable, even when *all* trustees collude, with a simple additional check on the final output [84].

3.2 Preliminaries

3.2.1 Related work

The blockchain literature has explored MPC in several regards. Perhaps the closest to Absentia is Enigma [120] which offers stateful MPC as a service. The original academic proposal utilizes a custom blockchain. Now as a commercial project, the emphasis is on providing generic smart contracts with privacy. Enigma runs on a Cosmos/Tendermint-based chain, with an Ethereum bridge contract that allows swapping crypto-assets. Ekiden uses TEE with blockchain to provide smart contracts that preserve confidentiality [31]. Absentia is different in the following regards: (1) users provide the circuit they want evaluated, (2) Absentia does not use trusted execution environments (TEE), and (3) we benchmark running natively on Ethereum. Like Enigma, Hawk also provides a privacy wrapper for

contracts [77] based on succinct zero knowledge. A fair MPC is described as an application of Hawk but not implemented. As an alternative to the current setting of Absentia using blockchain, a TEE could be used to provide verifiable computations. In fact, we effectively propose this in Chapter 6 with an additional property that the computation can only be performed once. Both systems are based on different trust models. We will provide a comparison of different approaches to MPC in Chapter 7.

The literature has also explored moving computation off-chain while not losing privacy or correctness, however from the perspective of a single entity's secret data (*i.e.,* verifiable computing as opposed to MPC). Examples include Zexe [18], ZkVM [5], and Raziel [97]. Another research direction, initiated by Andrychowicz *et al.* [6], explores how blockchain technologies can support an off-chain MPC to provide fairness. By contrast, Absentia is performing the MPC on the blockchain. Closely related to MPC are zero knowledge proofs, whose uses in blockchain are now too prolific to adequately summarize here.

3.3 Absentia: system design

High level flow. Figure 1 illustrates a high level overview of how participants interact with Absentia. The main contract of the system is the Absentia-DApp (`mixmatch.sol`), which can create sub-contracts: PET Sub-DApp (`PET.sol`). Note that Figure 1 is stylized and the exact implementation might split/join certain function calls but it provides an accurate mental model of participation within the system.

At the beginning of the protocol, the contracts are deployed, identifying Alice, Bob, and the trustees (by Ethereum address). Alice and Bob both submit their encrypted input, and deposit fees that will be paid to the trustees for completing the protocol. We consider Absentia *submit-and-go* because Alice and Bob do not have to perform any other functions during the execution of the protocol.

Certain tasks are public operations that can be performed by anyone. For our analysis,

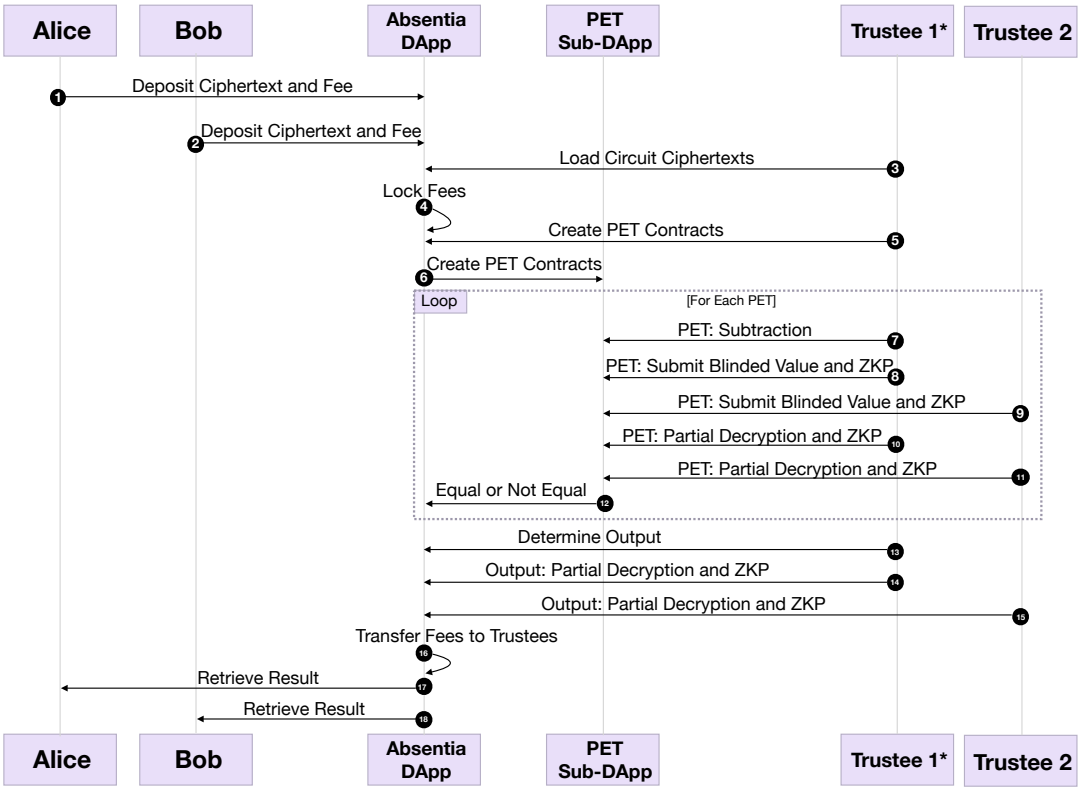


Figure 1: Overview of Absentia

we assume that Trustee 1 is the leader (denoted Trustee 1* with an asterisk) and always does these tasks. It is substantially more work, so it might improve the protocol to balance these operations between trustees or to compensate the leader more than the other trustees.

Recall Mix and Match protocol from Chapter 2. Mix and Match uses a partially homomorphic encryption scheme; we instantiate it with additive exponential Elgamal [38]. We describe later how implementing this protocol over the elliptic curve `secp256k1` (which is used natively by Ethereum) results in savings. For simplicity in Absentia, when we evaluate a circuit, we decrypt publicly if it is a final output. We also assume that circuits have been pre-computed and verified. In the future we may extend Absentia to accept a circuit and complete set of proofs to verify its correct construction, but for this chapter, we

concentrate on building a verifier for the online phase.

The actual Mix and Match operations done by each trustee is done off-chain using their share of the private key and other secrets (like randomizers) which are always offline. Ethereum is used to record the output of each step, record a zero-knowledge proof that the step was performed correctly, and to actually validate this proof. In this work, we cover every step with ZKPs and enforce an order among trustees, as there is a possibility of introducing vulnerabilities due to not enforcing a proper state machine. Enforcing this order also helps with identifiable aborts. Future work can explore whether trustees can work concurrently and/or aggregate proofs. The DApp will reject any outputs accompanied by invalid or incomplete proofs. All proofs are Σ-protocols (specifically Schnorr [98] or Chaum-Pedersen [27]) made non-interactive with (strong [14]) Fiat-Shamir [55]. As this is not our main contribution, we refer the reader to the original paper by Jakobsson and Juels for the full details how these proofs are used in Mix and Match [68].

For each gate, the Absentia DApp creates enough instances of the PETs (*e.g.,* 8 instances for a binary gate) to perform the evaluation. The trustees then interact with the PET contract, running each to completion (a state machine governs each step of the protocol). Note that for simplicity, Absentia requires the trustees to go in a specified order but the underlying protocol is amenable to some concurrency. If the output wires of the gates are inputs to another, trustees have to evaluate the gates in order. If not, then they can do the evaluation of the circuit in parallel. Once enough PETs are complete that the output is determined, the leader can assert this to the Absentia DApp which will check the state of the PET contracts to confirm. The final output is staged for decryption by the trustees. Alice and Bob can find it on the Absentia DApp. For simplicity, the result is in plaintext however Absentia could be modified to support proxy re-encryption instead of decryption which would leave two final ciphertexts, encrypted respectively under public keys specified by Alice and Bob.

Payments. Absentia allows Alice and Bob to pay Trustee 1 and Trustee 2 upon completion of the protocol. We implement a simple proof-of-concept payment scheme while noting more elaborate schemes are possible. As implemented, Alice and Bob can deposit and withdraw ETH. The protocol cannot begin until their accounts hold enough to satisfy the fee (and if they hold more, the excess can be withdrawn at any time). Once the protocol begins, the funds for the fee are locked in escrow within the contract. If the protocol reaches finality, the funds are transferred to the accounts of Trustee 1 and 2 who can then withdraw (Note we use standard re-entrancy protection[1] on withdraws.) If the protocol times out without reaching finality, the fees are returned to Alice's and Bob's accounts.

An alternative incentive scheme might pay trustees gradually for each step of the protocol they complete and then a larger bonus for completing. Since Absentia can identify which trustee aborts the protocol (a useful feature that is not always possible in MPC protocols), trustees could also be required to post a payment (stake) to act as a fidelity bond. They financially commit to completing the protocol in a timely fashion and their stake is taken (slashed) if they do not.

Code layout. Absentia is implemented in Solidity. All our code and tests are open source.[2] The trustees can perform their operations and generate their proofs in a language of their choice; we implement this in Mathematica (which we also use to generate test vectors for validating the Solidity code). `Mixmatch.sol` and `PET.sol` consists of 214 and 388 lines (SLOC) of Solidity code respectively. We adapt a standard library for elliptic curve operations.[3]

One optimization we implement concerns scalar multiplication over elliptic curves. Since Solidity is verifying proofs in Absentia, it only has to verify multiplications rather than perform them. Put another way, the trustee supplying the proof to Absentia already

[1] Open Zeppelin's ReentrancyGuard.sol
[2] `https://github.com/MadibaGroup/2017-Absentia`
[3] Orbs' ECops.sol

Code	Size(bytes)
bytecode	27,178
deployed	26,774
initialisation and constructor code	404

Table 1: Code size for mixmatch.sol

knows what the result of every multiplication is and can provide these values. As it turns out, it is cheaper to verify a multiplication than compute one by 'abusing' Ethereum's relatively inexpensive opcode for validating ECDSA signatures.[4]

Since Absentia generates a lot of PETs to perform the protocol, we implement this aspect with a factory design pattern. In this pattern, each PET is a stand-alone contract. The Mix and Match contract can create instances of these PET contracts and deploy them at new addresses. Our measurements (see below and Table 2) demonstrate that the factory pattern has certain drawbacks. `Mixmatch.sol` must deploy with a full copy of `PET.sol`'s bytecode in order for it to deploy instances of `PET.sol`. This results a contract size that is large. Also the function (Create Row) that creates (two) PETs each time it is called is the most expensive function in the system and costs $8,741,453$ gas (gas is Ethereum's metric for the cost of a computation).

All contracts enforce the order in which the functions can execute through state changes maintained within the contract. Key state changes emit events.

3.3.1 Measurements

Testing platform. To test Absentia, we use Truffle on a local Ethereum blockchain. Our test files are included on the code repository. We also duplicated Absentia's functionality in Mathematica to help establish correctness.

[4] V. Buterin, 2018. You can *kinda* abuse ECRECOVER to do ECMUL in secp256k1 today

Code size. The code size for `mixmatch.sol` is outlined in Table 1. When any Ethereum contract is first deployed, the constructor can only be run once. Thus the constructor code does not need to be referenced for further invocations and is not stored with the deployed bytecode (but can be found in the calldata of the deployment transaction).

When compiled, `mixmatch.sol` is $26,774$ bytes (plus a constructor of 404 bytes). Because of the factory design, this includes the bytecode to create `PET.sol` contracts. Ethereum limits contracts to ≈24KB (per EIP170).[5] We simply adjust Truffle's limit to allow us to benchmark it as a single contract. However it cannot be deployed on Ethereum today as is. Straightforward options to bring the code under the limit include: (1) taking `PET.sol` out of the contract and having the leader deploy each PET contract and load the addresses back into `mixmatch.sol`; (2) move stateless functions to libraries; (3) split the contract up arbitrarily and use `delegatecall` to execute the pieces in a common context; or (4) find ways to optimize the code to reduce its size (it is academic, proof of concept code, and is very close to the limit, so this should be feasible).

Gas costs. Table 2 provides the cost to deploy Absentia's two contracts and one library, as well as the gas costs of each function. Note that many functions are invoked more than once in a complete run of Absentia. The gas costs are as reported in Truffle's local network (Ganache). To convert gas into USD, we use 1 gas = 87 Gwei as recorded on Dec 01, 2020.[6] The price of ETH is $\$615.07$ for the same date.[7]

As the leader of the protocol, Trustee 1 (T1*) has to perform more operations than the other participants. Table 3 shows the costs per participant. Particularly expensive tasks for the leader is loading all the ciphertexts for the circuit into the contract and initializing the memory needed, in particular for each PET, for the working memory. This is why,

[5] In 2016 when EIP170 was finalized, a 24KB contract could not deploy without crossing the block gas limit, however the gas limit has increased substantially since.

[6] Etherscan

[7] Coinmarketcap

Contract	Function	Gas	Gas Cost ($)
ec.sol	Deploy Contract	595,517	31.94
Mixmatch.sol (Absentia DApp)	Deploy Contract	6,091,398	326.75
	A&B: Load Funds	28,040	1.50
	T1*: Load Outputs	300,798	16.13
	T1*: Create Row	8,741,453	468.90
	T1*: Find Matching Row	37,547	2.01
	T1*: Find Matching Value	40,868	2.19
	T1*: Create Final Decryption	4,430,611	237.66
	A&B: Withdraw Excess Funds	41,110	2.21
	A&B: Withdraw Funds	39,221	2.10
PET.sol (PET Sub-DApp)	Deploy Contract	4,681,858	251.14
	A&B or T1*: Load Ciphertexts	304,668	16.34
	T1*: Subtraction	242,131	12.99
	T1*: Randomization ZKP	815,340	43.74
	T2: Randomization ZKP	393,561	21.11
	T1*: Partial Dec ZKP	364,298	19.54
	T2: Partial Dec ZKP	363,612	19.50
	T1*: Full Decryption	107,086	5.74
	T1*: Load Final Ciphertexts	173,945	9.33

Table 2: Gas costs per function and who runs the function: Alice (A), Bob (B), Trustee 1 as the leader (T1*), or Trustee 2 (T2). Note that many functions are run more than once.

	Alice	Bob	Trustee1*	Trustee2
Number of Transactions	5	5	44	17
Total gas cost	1,246,712	1,246,712	52,952,603	6,420,996
Total cost in USD	66.87	66.87	2840.41	344.43

Table 3: Cost for each participant

for example, Randomization ZKP is so expensive for T1 as compared to T2 (the code of both functions is identical but gas costs are $815,340$ versus $393,561$). Trustee 1 initializes many state variables (more expensive in Ethereum) that are not needed once the function completes; while trustee 2 overwrites the variables (less expensive in Ethereum). The next function, Partial Decryption, continues overwriting these variables.

Our design has some room for improvement. For example, in the current implementation, Alice and Bob have to deposit their inputs for each PET contract that is created (8 in total). A better design pattern (more consistent with Figure 1) would have Alice and Bob

Setting	Total gas
1 gate, 2 trustees	61,867,023
2 gate, 2 trustees	121,240,622
1 gate, 3 trustees	68,288,019

Table 4: Cost of scaling Absentia

deposit once in `mixandmatch.sol` and have the factory contract initialize the PETS with the correct values. Another improvement would aim to reduce the total transaction count for each participant by merging operations that are performed in a sequence by the same participant (we split them into logic blocks to better showcase what the gas was being spent on).

In Table 4, we show how Absentia scales with additional gates and additional trustees. If we want to evaluate a two gate circuit, Alice and Bob still perform the same number of transactions but nearly all of the rest of the functions are run twice as many times. Note that if the output of one gate is fed into the next gate, the leader (T1*) will load the inputs for the second gate. Going back to a single gate, increasing the number of trustees from 2 to 3 is not as expensive. Each additional trustee has a marginal cost equal to Trustee 2's cost in Table 3.

3.4 Absentia on Layer 2

3.4.1 Roll-ups

A loose collection of technologies, called *Layer 2* solutions, have been proposed to address certain shortcomings of operating directly on Ethereum (*Layer 1*) or other blockchains [61]. Different approaches to scaling blockchain using rollups have been explored in the literature [105]. These solutions generally strive for one or more of the following: reducing transaction latency, increasing transaction throughput, or reducing gas costs. In the case of Absentia, reducing gas costs is paramount. However Layer 2 solutions can also change the

threat model; for Absentia, we require that Alice and Bob can trust the final output without having to verify any proofs themselves.

The most appropriate layer 2 technology for our requirements is called a *roll-up* which targets gas costs. In Ethereum, every transaction is executed (and thus validated) by every Ethereum node. In a roll-up, transactions are executed by off-chain nodes called *validators*. Validators try to convince the Ethereum network that the result of the transaction execution (*i.e.,* the state change of the EVM) is correct without the Ethereum nodes having to execute it.

Since Ethereum nodes cannot just ignore the Ethereum protocol's specifications for how to validate transactions, the roll-up cannot be implemented on Layer 1. Rather it is implemented inside its own DApp (Layer 2). This Layer 2 DApp is effectively a container, operating by its own custom consensus rules, for DApps that want roll-up functionality. The tradeoff is they are isolated from regular L1 DApps without some additional protocols (*e.g.,* interoperability support for currency/token transfers and external function calls). For Absentia, we do not require interoperability with L1 other than having a currency in L2 to pay the trustees.

There are at least two ways to convince on-chain participants that an off-chain computation was performed correctly. The first is to prove it with a succinct proof. SNARKs are one proof-type for general computations that are more efficient to verify than performing the computation itself. A second approach (called an *optimistic rollup*) is to have a validator assert the result and then allow for anyone to dispute it before finalizing it. Resolving disputes is always possible by having the Ethereum nodes perform the computation itself, but disputes can be settled in a more succinct way (see [71]). If Alice demonstrates that a validator is wrong, the validator is financially punished and Alice is rewarded. Such validators do less work than Ethereum nodes (as well as validators that have to produce SNARKs)—therefore, optimistic rollups enable substantially lower gas costs.

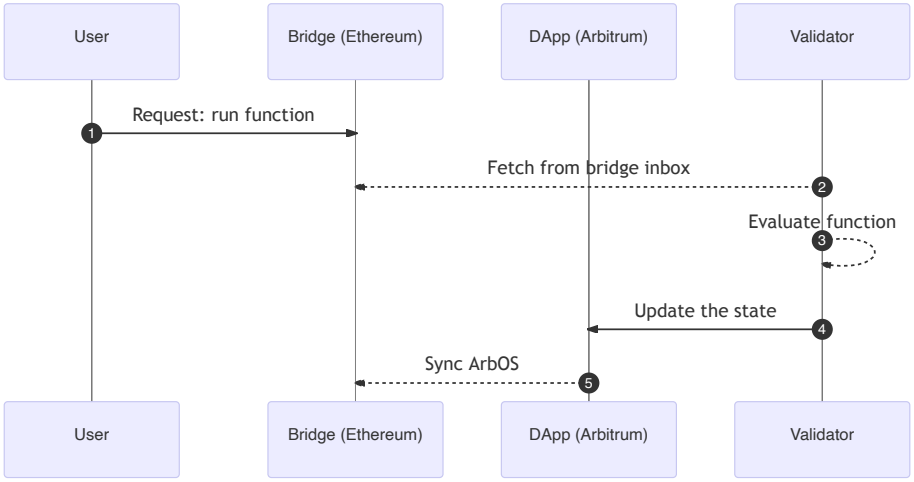


Figure 2: Overview of Arbitrum transaction submission

3.4.2 Arbitrum

Arbitrum is a Layer 2 solution proposed in a *USENIX Security* paper [71] and now maintained as a commercial project by Offchain Labs. Currently, they operate an optimistic rollup on Ethereum. Instead of operating all Arbitrum contracts (called *ArbOS*) in a container DApp on Ethereum, ArbOS instead operates as a side-chain. A *bridge contract* on Ethereum serves as an interface between Ethereum and Arbitrum. Figure 2 shows how function calls work on Arbitrum. A user initiates a transaction on Ethereum to the Bridge Contract with the instruction to deploy a contract or run a function, along with all the data required for Arbitrum to perform this transaction. A validator sees new transactions in the inbox of the bridge, executes one and asserts the result to ArbOS. After a dispute period, the transaction is considered finalized. Periodically, the entire state of ArbOS is committed back to Ethereum. As all Arbitrum transactions are recorded on Ethereum, anyone can compute and compare the current ArbOS state.

	Ethereum		Arbitrum			
Function	**Tx**	**Gas**	**L1 Tx**	**L1 Gas**	**L2 ArbGas**	**Size**
Deploy ec	Link	1,103,372	Link	80,152	1,304,481	4978
Deploy PET	Link	5,266,352	Link	386,079	4,260,273	24,172
Load Ciphertexts	Link	305,309	Link	7869	820,507	742
Subtraction	Link	260,729	Link	5469	4,789,799	550
T1 Randomization ZKP	Link	819,877	Link	11,488	10,972,720	644
T2 Randomization ZKP	Link	398,245	Link	11,440	11,069,485	742
T1 Partial Dec ZKP	Link	366,636	Link	11,452	10,692,786	742
T2 Partial Dec ZKP	Link	366,089	Link	11,512	10,689,113	742
Full Decryption	Link	124,816	Link	6236	4,258,675	422

Table 5: Comparison between deploying a plaintext equality test on Ethereum and deploying on Arbitrum (via Ethereum). The links show the reader the actual transactions of a test-run on Kovan/Arbitrum's respective block explorers. Size is the calldata in bytes. Note that we do not include the links to the L2 transactions, as the transactions recorded on their test net was removed by the project.

3.4.3 Absentia on Arbitrum

Testing platform. Arbitrum runs a testnet with a bridge on Ethereum's Kovan testnet. As mentioned above, Absentia is too large to deploy (as a factory contract) within Ethereum's contract size limit. To experiment with Arbitrum, we implement only the PET sub-module as a standalone contract. We run the tests with Truffle. Instead of sending transactions to the Arbitrum bridge (as in Figure 2), Arbitrum runs a service for developers where transactions are sent (off-chain) to a relay server (called an *Aggregator*) which will batch all pending transactions together as a single Kovan transaction to the bridge (and pay the gas). However we report the measurements as if the participants were sending the transactions themselves.

Gas costs. Table 5 compares the cost of running a plaintext equality test (PET) on Ethereum (specifically Kovan testnet) and running it on Layer 2 (L2) with Arbitrum. Note the Ethereum numbers differ slightly from Table 2 as it is deployed on a different testnet (Kovan instead of private) and we modified it slightly to be a stand-alone DApp.

Arbitrum creates two transactions (recall Figure 2): the Ethereum gas cost of relaying

the (layer 1 or L1) transaction to the Arbitrum bridge, and the cost for the validator to execute the function, measured in ArbGas. The cost of the first Arbitrum transaction (*L1 Gas*) is paid with ETH but is invariant to its computational complexity. It is essentially only a function of its size (compare *L1 Gas* to *Size*). Note that the gas costs listed on the Kovan block explorer (links under *L1 Tx*) are for aggregated batches of transactions. We report what the cost would be to send it directly (not through an aggregator).

The ArbGas cost on Arbitrum should be similar to the gas cost on Ethereum, however validators do not run EVM bytecode directly. It is translated into Arbitrum virtual machine (AVM) bytecode which has its own opcodes and ArbGas costs. ArbGas has no market price currently. It is expected to be much cheaper than Ethereum's gas. In practice, the trustees could act as validators for Absentia transactions as they have to perform the computation anyways. Therefore we approximate ArbGas as free.

A run of PET on Ethereum costs 9,011,425 gas (or 483.38 USD), while on Arbitrum the cost is 531,697 gas (or 28.52 USD). In this use case, Arbitrum reduces Ethereum gas costs by 94%. Even though there is a significant decrease in the cost, it is still expensive to evaluate bigger gates. However, we believe that our proposed system serves as a starting point for future work to address decreasing the costs.

3.5 Concluding remarks

Ethereum can complement secure function evaluation protocols by enabling coordination, providing incentives, and enforcing correctness. Given recent developments in Ethereum toward performance and scalability, we felt it was an appropriate time to benchmark how expensive MPC is on Ethereum. Even though we expected it to be expensive, we did not imagine a single binary NAND gate would cost thousands of dollars on Ethereum. Most 'interesting' circuits are probably at least hundreds of gates, with many applications that would require many orders of magnitude more.

Despite this, we argue that Absentia is still an important research contribution. It proves the concept works, establishes a lower bound, and it sets a new research challenge: through improvements, how many gates can be evaluated for, say, under $100 USD? Today it might be less than one but we are confident that future research can improve that number substantially. For example, our code can be further optimized; the latest MPC techniques can be applied; and Σ-protocols can be replaced with succinct zero-knowledge proofs. Meanwhile, Layer 1 and Layer 2 technologies will continue progressing, and faster L1 solutions can be explored as a future work. If we were to redo the project, we might also look at newer technologies like Avalanche, which is a faster L1 solution.

In this chapter, we presented Absentia, that provides submit-and-go, public verification and identifiable abort functionalities in a fully malicious setting. The system does not require an additional trusted hardware. We presented a two-party setting in this chapter, but it can be extended to three or more parties. Absentia is less cost efficient and has lower performance compared to the works that we will explain in the upcoming chapters. We provide a comparison framework with other approaches to MPC in Chapter 7.

Chapter 4

Private set intersection for link prediction

This chapter is based on the work supervised by Erman Ayday and Jeremy Clark and co-authored with Mina Namazi and it is published at 2022 Workshop on Data Privacy Management (DPM) co-located with European Symposium on Research in Computer Security (ESORICS) [48].

In Chapter 3, we introduced a blockchain-based approach to MPC. Currently, it is costly to utilize Absentia for bigger circuits with a large number of gates. In this chapter, we look at a specific type of MPC, namely private set intersection cardinality, for performing privacy-preserving link prediction between two different graph databases.

4.1 Introduction

Link prediction discovers important linkages between nodes in a graph. Based on the analysis of these linkages, it helps the data holder to forecast what future connections might emerge between the nodes, and to predict if there are missing links in the data. Some common applications include: (i) in social networks, to recommend links between users;

(ii) in e-commerce or personalized advertisement, to recommend products to users; (iii) in telecommunication, to build optimal phone usage plans between the users; and (iv) in bioinformatics, to predict associations between diseases and attributes of patients or to discover associations between genes (or proteins) and different functions.

Link prediction is typically done on the local graph of a data holder or service provider. For instance, a social network, analyzing the common neighbours between its users decides whether to recommend links between the users. However, link prediction will be more accurate and correct by considering more information about the graph nodes. This can be achieved by merging two or more graph databases that include similar information, leading to "distributed link prediction" between two or more graph databases. For instance, two social networks may utilize the connections in their combined graph to provide more accurate link prediction for their users. Furthermore, distributed link prediction will enable different uses of link prediction, such as building connections between users and products based on the tastes of other similar users (*e.g.,* friends of a user). Such an application may be possible between graph databases of a social network and an e-commerce service provider. In some cases, collaboration is mutually beneficial to both parties. In others, one party can pay the other party to participate—one party gets better data and the other gets to monetize its data.

Distributed link prediction, although is a promising approach for more accurate and richer link prediction applications, also results in privacy concerns since it implies combining two or more different graph databases. In this scenario, threats against privacy can be categorized into three groups [110]: identity disclosure, link disclosure, and attribute disclosure. All these threats should be considered in a distributed link prediction algorithm, since it involves privacy-sensitive databases from multiple parties.

One promising solution for this privacy concern is cryptography to achieve distributed link prediction in a privacy-preserving way. Thus, in this paper, our goal is to develop

a cryptographic solution for privacy-preserving distributed link prediction between multiple graph databases. We propose a solution based on private set intersection cardinality (PSI-CA) to tackle this problem by considering both the efficiency of the solution and its privacy. Via evaluations, we show that this solution provides good efficiency. For example, it can run in under 1 second (ignoring communication latency) for graphs based on a Flickr dataset with 40K nodes. The proposed protocol does not provide perfect privacy (it leaks some intermediary values) and so we quantify this leakage to better understand if it is consequential enough to move to a fully private solution (which we also sketch).

4.1.1 Use Cases

Privacy-preserving distributed link prediction can be utilized in different settings. Here, we explain some of the possible applications.

Social networks. In this setting, there are two social networks, Graph 1 and Graph 2. Graph 1 aims to understand whether there will be a link formed between nodes x and y by also utilizing the similarity of x and y in Graph 2, as distributed link prediction provides better accuracy compared to performing this operation locally.

E-commerce. Another application can be between a social network and an e-commerce service. In the previous use case, the link between two users is the main concern of the protocol. Unlike the previous case, here the links between a user and products are determined at the end of the protocol. In the e-commerce graph, there are links between the users and the products that they have bought. The aim here is to provide better advertising to users. The network will recommend product n to the user x if this user's friends also purchased the same product. For this purpose, the e-commerce network has to know the friends of user x in the social network. Unlike the previous use case, here link prediction cannot be done locally on the e-commerce graph, as the knowledge of the social network's structure

should be utilized in order to do the recommendation.

Telecommunication. In this use case, an advertising company wants to propagate an advertisement in the telecom network. If user x is a target for that advertisement, the company would like to know which nodes are likely to form links with user x, in order to decide which nodes it will send the advertisement. The aim is to maximize the number of nodes that learn about the advertisement. Another application involves a social network graph and a phone operator graph. The phone operator wants to find out friends of user x in the social network, so that it offers the special services (e.g., discounts) to the users that are similar to user x.

Bioinformatics. Here, the first graph consists of patients and diseases and the aim is to predict the link between the patient i and the disease j. In the second graph, there are similar patients to patient i. Using these similar patients, and their connection to disease j, the link between patient i and disease j can be inferred.

4.1.2 Related Work

There is a rich literature on link prediction algorithms (without consideration of privacy) in a variety of network structures: multiple partially aligned social networks [116]; coupled networks [51]; and heterogenous networks [104]. Other works consider node similarity when two nodes in the graph do not share common neighbours [79]; unbalanced, sparse data across multiple heterogeneous networks [50]; missing link prediction using local random walk [82]; and the intersection of link prediction and transfer learning [114].

Other research considers the use of cryptography for collaborating on graph-based data between two parties with privacy protections. However such works consider problems other than link prediction: merging and query performed on knowledge graphs owned by

different parties [28]; whether one graph is a subgraph of the other graph [112]; single-source shortest distance and all-pairs shortest distance both in sparse and dense graphs [4]; all pairs shortest distance and single source shortest distance [19]; and transitive closure [64]; anonymous invitation-based system [16] and its extension to malicious adversarial model [17]. While it may be possible to transform some of these into finding common neighbours with a black-box approach, we provide a purpose-built protocol for common neighbour. Later in Section 4.2.3, we review potential cryptographic building blocks in the literature.

Finally, there is one work that focuses on using cryptography specifically for link prediction in order to provide privacy protections. Zhang *et al.* present a protocol for privacy-preserving link prediction [115]. This paper presents a more general framework for link prediction, while ours is tailor-made for common neighbours. If the setting is known ahead of time (*i.e.,* common neighbours will be used), our protocol can be adopted. However, it is difficult to adapt our protocol to other metrics. Note that, we were not aware of this paper at the time we were working on this project.

4.2 Proposed Solution

4.2.1 Building Blocks from Data Mining

Link prediction. Given a snapshot of a graph at time t, link prediction algorithms aim to accurately predict the edges that will be added to the graph during the interval from time t to a given future time t' [80].

Similarity metrics. In Table 6, different metrics for calculating proximity are given. Common neighbours, Jaccard coefficient and Adamic-Adar index are regarded as the node-dependent indices and they only require the information about node degree and the nearest

Similarity metric	**Definition**
Common neighbours	$\lvert\Gamma(x) \cap \Gamma(y)\rvert$
Jaccard's coefficient	$\frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$
Adamic/Adar	$\sum_{z\epsilon\lvert\Gamma(x)\cap\Gamma(y)\rvert} \frac{1}{log(\lvert\Gamma(z)\rvert)}$
$Katz_\beta$	$\sum_{l=1}^{\infty} \beta^l . \lvert path_{x,y}^{\langle l\rangle}\rvert$ where $path_{x,y}^{\langle l\rangle} :=$ { paths of length exactly l from x to y} weighted: $path_{x,y}^{\langle l\rangle} :=$ weight of the edge between x and y unweighted: $path_{x,y}^{\langle l\rangle} := 1$ iff x and y are 1-hop neighbours The weight is determined by the constant value β.

Table 6: Different similarity metrics in a graph.

neighbourhood, whereas the Katz index is defined as path-dependent index that consider the global knowledge of the network topology [82]. While there are also other metrics that are used (some of which are shown in Table 6), we choose common neighbours, as it is one of the widely-used methods for link prediction.

Common Neighbours. Common neighbours is used to predict the existence of a link between two nodes based on the number of their common neighbours. If two nodes share common neighbours, it is more likely that they will be connected in the future. In a local graph, the result of the metric can directly be computed by determining the intersection of the neighbour sets of two nodes. Based on the cardinality of the set, the network decides whether to suggest a link between these two nodes. The cardinality is defined as

$$\text{common neighbours} = \lvert\Gamma(x) \cap \Gamma(y)\rvert$$

, where $\Gamma(x)$ and $\Gamma(y)$ are the set of neighbours of nodes x and y respectively. For example, when user x signs up, the graph will complete the protocol and show the user recommendations in descending order of the common neighbours cardinality. We assume that the bigger the cardinality is, the more probable that the two nodes will become neighbours.

Adapting Common Neighbours for Two Parties. For the use case in this paper, we look at the problem of computing common neighbours metric across two different graphs owned by different entities. For example, this could be two separate social networks, or a social network with an e-commerce network. Graph 1 wants to perform link prediction between the nodes x and y by using the common neighbours information from Graph 2. CN denotes the total number of common neighbours that will be determined at the end of the protocol. We propose the following computation for two graphs, taking care to not double count any common neighbours:

$$\mathrm{CN} = \mathrm{local1} + \mathrm{local2} + \mathrm{crossover1} + \mathrm{crossover2} - \mathrm{overlap}$$

The variables are as follows:

- local1: number of common neighbours of node x and node y in Graph 1
- local2: number of common neighbours of node x and node y in Graph 2
- crossover1: number of common neighbours of node x from Graph 1 and node y from Graph 2
- crossover2: number of common neighbours of node y from Graph 1 and node x from Graph 2
- overlap: intersection of local1 and local2

Figure 3 illustrates an example of how CN is computed using the neighbours sets of both Graph 1 and Graph 2 (based on the graphs in Figure 8). Graph 1 decides whether to suggest a link between nodes 1 and 6 based on this cardinality.

PSI based	PSI [42]	**Complexity:** Protocol complexity is linear in the sizes of the two sets. Both the client and the server performs exponentiations and modular multiplications. **Info leaked:** Intersection cardinality, no third party **Security setting:** semi-honest
	Delegated PSI [52]	**Complexity:** Computation and communication complexity of the protocol is linear in the size of the smaller set. For polynomial interpolation field operations are performed. Cloud server has to evaluate oblivious distributed key PRF instances, and unpack messages. The waiting time of packing messages by the backend server is the main computation cost. **Info leaked:** Intersection cardinality, uses third party **Security setting:** semi-honest
	PSI with FHE [30]	**Complexity:** Communication overhead is logarithmic in the larger set size and linear in the smaller set size. While FHE is asymptotically efficient, it isn't in practice. **Info leaked:** Input sizes and bit string length of the sets, no third party **Security setting:** Semi-honest
	PSI with OT [93]	**Complexity:** The circuit-based PSI protocol has linear communication complexity. **Info leaked:** No info leaked as the result of a function on the intersection cardinality is the output, no third party **Security setting:** semi-honest
	Labeled PSI with FHE in malicious setting [29]	**Complexity:** Communication overhead is logarithmic in the larger set size and linear in the smaller set size. While FHE is asymptotically efficient, it isn't in practice. **Info leaked:** No info is leaked, as the output is secret shared, no third party **Security setting:** Malicious
Non-PSI based	Privacy-preserving integer comparison [67] over each pair	**Complexity:** Privacy-preserving integer comparison protocol is run between every pair of nodes in the adjacency matrix created using the neighbour list from both graphs. The comparison protocol performs encryption, partial decryption, modular exponentiation, and multiplications. **Info leaked:** No info is leaked, no third party **Security setting:** Malicious if ZKP added

Table 7: PSI-based and Non-PSI based cryptographic building blocks

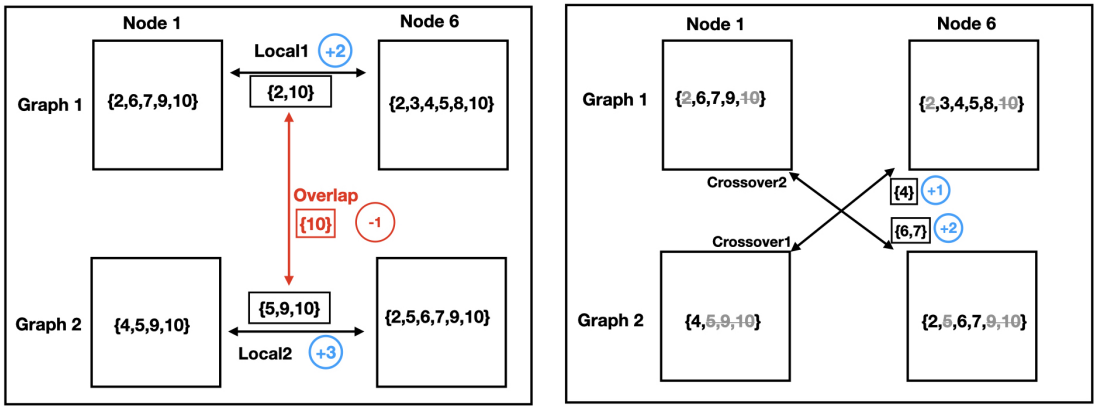


Figure 3: An example computation between Graph 1 and Graph 2 to find the number of common neighbours of nodes 1 and 6 in their joint graph. Our contribution is to perform this computation in a privacy-preserving manner.

4.2.2 System Model

In our setting, there are two parties: Graph 1 and Graph 2, each having a graph structured network. Graph 1 wants to determine whether to suggest a link between the nodes x and y, not by only determining the common neighbours using its own graph, but also utilizing the graph structure of Graph 2. Graph 1 and Graph 2 compute common neighbours on their joint graphs without disclosing their respective graph structures. The result (number of common neighbours) is provided only to Graph 1, however the protocol can be run twice if Graph 2 also wants the result (otherwise, we assume Graph 1 is paying Graph 2 for this service). While creating our scheme, we make the following assumptions:

1. The identifiers in both graphs for the same nodes match. The graphs should prepare for the computation by sharing a schema and agreeing on unique identifiers (*e.g.,* an email address or phone number for human users).

2. Both graphs know the identity of the nodes for which the computation is being performed. In other words, edges involving these nodes are hidden, as well as all other edges and nodes.

3. If x and y are direct neighbours in Graph 1, Graph 1 has no need for the computation.

4. If x and y are direct neighbours in Graph 2, Graph 2 will halt before doing the computation and inform Graph 1. In this case, Graph 1 discovers a hidden link between x and y, which is stronger for prediction than the number of common neighbours.

Threat Model. The common public input to the computation will be the identifiers of two nodes known to Graph 1 and Graph 2. The private input of Graph 1 and Graph 2, respectively, is an assertion of their graph data. We assume Graph 1 and Graph 2 honestly input their correct data. This is a common assumption and resolving it involves having the data authenticated outside of the protocol, which is not a natural assumption for our use-cases. The second question is whether we can assume Graph 1 and Graph 2 follow the protocol correctly (semi-honest model) or exhibit arbitrary behaviour (malicious model). Given the strong assumption of data input, we find it natural to fit it to a semi-honest model of the protocol.

With these assumptions, we design the protocol so that Graph 2 learns nothing about Graph 1 other than the common input. On the other hand, Graph 1 learns the number of common neighbours on the joint set, which is the output of the multiparty computation (MPC). A fundamental limitation of MPC is that the output itself can leak information about the input. For example, if Graph 1 is malicious and is able to repeat this protocol many times with Graph 2, it can slowly reconstruct Graph 2's input by adaptively providing different inputs each time. For the purposes of this paper, we assume Graph 1 will not do this, because it is semi-honest, and further Graph 2 would not entertain so many executions of the protocol.

As an artifact of our protocol, Graph 1 also learns the intermediate values to compute the number of common neighbours: local1 + local2 + crossover1 + crossover2 - overlap. This extra information does allow a malicious Graph 1 to reconstruct Graph 2 with fewer queries, but in Section 4.3.3 we show that the impact of the leakage is immaterial. This can be prevented with heavier cryptography (Section 4.4.1). In addition, Graph 2 can force

Graph 1 to compute the wrong result only if it behaves maliciously. In conclusion, our threat model provides reasonable privacy protection while being lightweight enough to be practical, and we suggest it represents a useful compromise for many real-world applications.

4.2.3 Building blocks from cryptography

Private set intersection (PSI) is a two-party cryptographic protocol that allows two entities, each with a set of data, to learn the intersection of their data without either learning any information about data that is outside the intersection [57]. After a detailed investigation of PSI variants and other related primitives, we choose [42] as the core scheme to deploy for our link prediction. Our scenario requires a scheme that calculates only the cardinality (called PSI-CA, recall from Chapter 2) of the intersection of the two sets in an efficient and scalable way with minimum information leakage with no third party's assistance. A security model for semi-honest adversaries is sufficient, and we leave additional (stronger) security guarantees for future work.

A summary of relevant cryptographic primitives is given in Table 7 and we provide further details in this section.

Comparison of cryptographic primitives. De Cristofaro [42] introduced a method to calculate PSI-CA in which the client or the server learns no information about each others' private input. Moreover, the two instances of the protocol on the same inputs are unlinkable. Beyond revealing the cardinality to the client, the upper bound of each set is the only information leaked to the parties. The protocol is secure against semi-honest adversaries who follow the protocol's instructions. The complexity remains linear in the size of the two sets.

In some scenarios, where the computation power of the parties is limited, they delegate

the computation to an untrusted cloud server. Duong *et al.* [52] suggested a delegated PSI-CA based on oblivious distributed key PRF (Odk-PRF). The protocol has two phases. In the first phase, the party who wants to receive the cardinality result (in our scenario, Graph 1) distributes their input's secret shares to the cloud server. The Odk-PRF protocol is deployed between the cloud server and the sender party (who is Graph 2). As a result, the cloud server obtains secret shares of the PRF output for each share of Graph 1's input, and Graph 2 learns the combined PRF key. In the second phase, Graph 2 generates a set of key-value pairs using their own inputs and Graph 1's input shares. The cloud server can obliviously search graph 2's key-value pairs and obtain correct values known to Graph 1. The server adds some fake values, permutes, and sends them to graph 1, which can check the number of items in the intersection (PSI-CA) by counting the "real" values obtained from the cloud server. The protocol is secure against semi-honest adversaries who can corrupt parties. Finally, the cardinality result is revealed to Graph 1, and the secret shares of Graph 1 are revealed to the cloud server. Moreover, as a result of running the Odk-PRF protocol with Graph 2, The cloud server views a subset of the PRF value. Graph 2 has PRF key shares. Although these values reveal no information about the parties' secret/key shares, the setting requires non-colluding parties. The computation and communication complexity of the protocol is linear in the size of the smaller set.

Chen *et al.* [30], developed a PSI protocol based on fully homomorphic encryption. Graph 1 holds a set X with the size of N_x, and Graph 2 has set Y with the size N_y. These sets consist of σ-bit strings. Graph 1 encrypts each element of its set and sends them to Graph 2 who homomorphically evaluates the product of differences of these values with all of the Graph 2's items. Graph 2 randomizes the results by multiplying the items with a uniformly random non-zero plaintext, and sends the result back to Graph 1. The latter decrypts the results to zero when there exists a common item in Graph 2's set. Otherwise,

the result returns a uniformly random non-zero plaintext. The security holds in the semi-honest model. This protocol leaks no information about the sets to the parties. Apart from revealing the intersection result to Graph 1, the set sizes, the (common) bit-length σ are publicly known values. The communication complexity of the protocol is linear in the size of the smaller set, and logarithmic in the larger set.

Pinkas *et al.* [93] developed a solution for PSI based on oblivious programmable pseudo-random functions (OPPRF). There protocol comprises of a circuit that its input wires are associated with the hash of input values in Graph 1 and Graph 2's sets. If the items in both sets are equal the circuit's output wire returns 1, and 0, otherwise. The circuit computes the number of 1's and reveals it. This solution [93] reveals the set intersection result and bit-length of the items to both parties. The security is modeled for malicious parties. Moreover, the complexity is linear.

Chen *et al.* [29] developed a labeled PSI-based solution based on OPRF and homomorphic encryption. In this setting, Graph 2 holds a label l_i for each item in its set. Graph 1 will also learn the labels corresponding to the items in the intersection. The parties use OPRF in a pre-processing phase to hash their values. Then, they deploy FHE to obtain the final result. The complexity of the scheme remains logarithmic in the size of larger set and linear in the size of smaller set. While the intersection result and a label are delivered to Graph 1, the number of items in each set after hashing is leaked to public. The security of the scheme is proven for malicious adversaries.

4.2.4 Proposed protocol

We use PSI scheme proposed in [42] to perform distributed link prediction between two graph databases. Figure 4 shows the interactive protocol between Graph 1 and Graph 2. Graph 1 wants to learn the common neighbour index to determine whether to suggest a link between the nodes x and y. Both Graph 1 and Graph 2 locally determine the neighbour sets

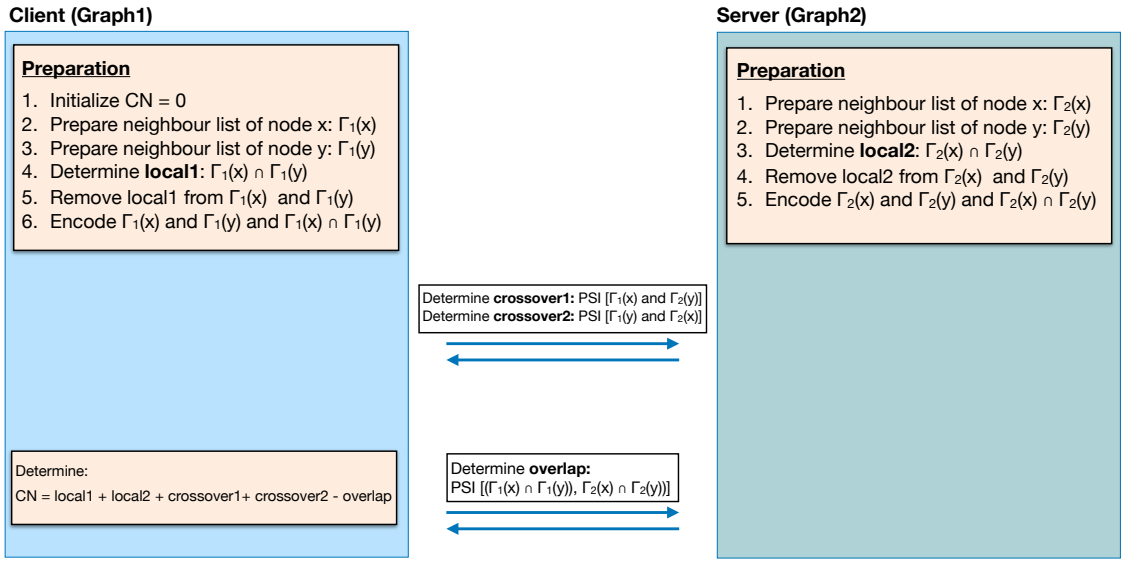


Figure 4: Overview of the proposed PSI-CA based solution. PSI-CA is called three times in the protocol for determining crossover1, crossover2 and overlap. PSI-CA itself is described in Figure 5

of x and y (local1 and local2, respectively). In order to determine crossover1, crossover2, and overlap, Graph 1 and Graph 2 run three separate PSI protocols among themselves. Each PSI leaks a certain amount of information and we discuss this partial information leak in Section 4.3.3. At the end of the protocol, Graph 1 learns the exact cardinality of common neighbours of nodes x and y on the joint graph. Figure 5 shows the details of the PSI protocol for calculating crossover1 (the calculations for crossover2 and overlap are also the same). It is an interactive protocol between Graph 1 and Graph 2, with offline and online stages. At the offline stage, Graph 1 masks its set and Graph 2 masks its set and shuffles it. During the online stage, Graph 2 receives the masked set of Graph 1, masks it with its own randomness and shuffles it. When Graph 1 receives the sets, it removes the randomness and determines the intersection of two sets. PSI is described in the multiplicative subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p^*$, where p and q are large primes, $q \mid p-1$ and $g \, \epsilon \, \mathbb{G}_q$ is the generator. $|p| = 1024$ bits and $|q| = 160$ bits. This is for experimental purposes only, these parameters should be at least doubled in length to meet the current, accepted security level [1]. H and H' are hash

[1]NIST Special Publication 800-131A Revision 2

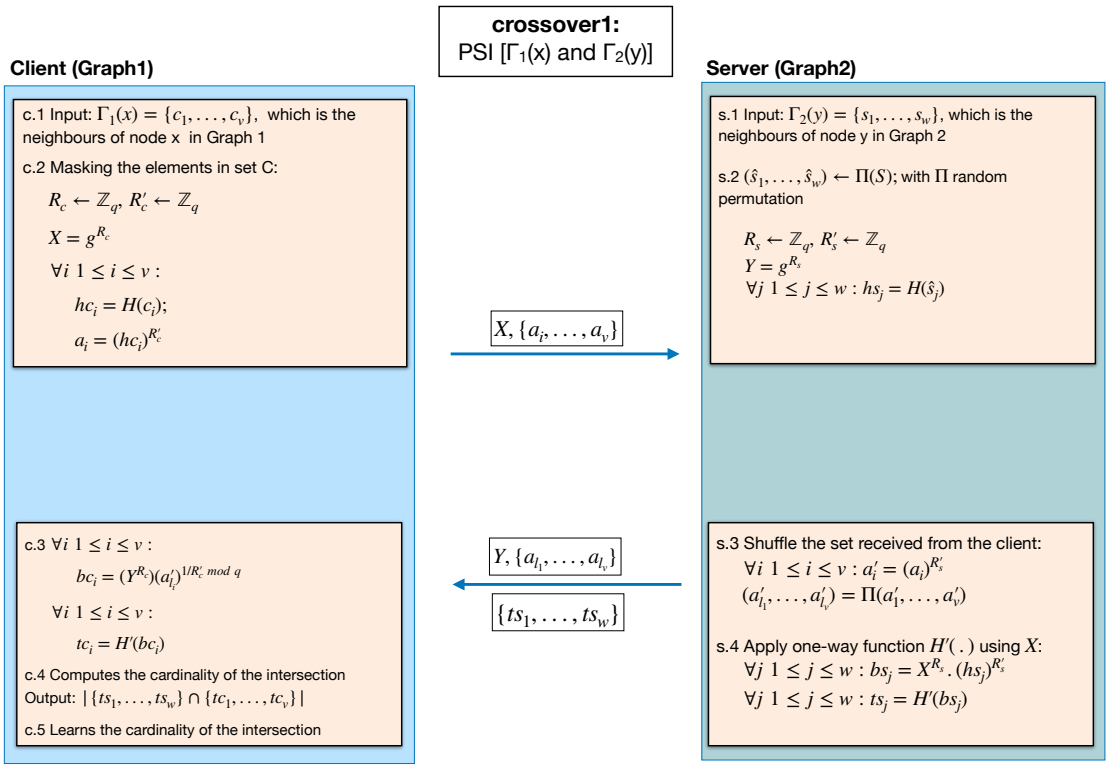


Figure 5: PSI-CA protocol for determining crossover1 (adapted from [42])

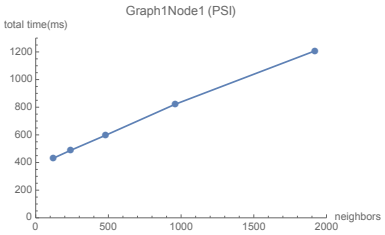
functions that are modeled as random oracles.

4.3 Evaluation

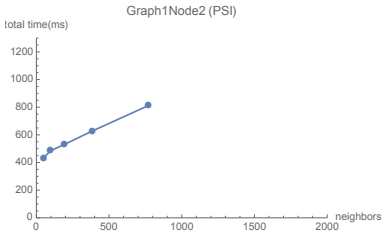
4.3.1 Performance

We implemented the proposed distributed link prediction algorithm and evaluated it considering different aspects. Our code is open source.[2]. We used the implementation of PSI defined in [42] where q and p are 160 and 1024 bits, respectively. We ran our experiments on macOS High Sierra, 2.3 GHz Intel Core i5, 8GB RAM, and 256GB hard disk. We ran each experiment for 20 times and reported the average. Note that, if we were to do the same evaluation today, we would compute the confidence interval.

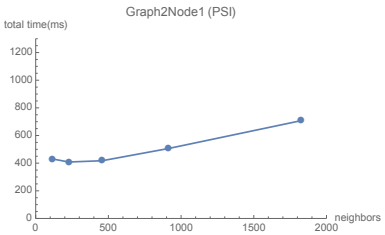
[2] https://github.com/ddm2/2022-LinkPrediction



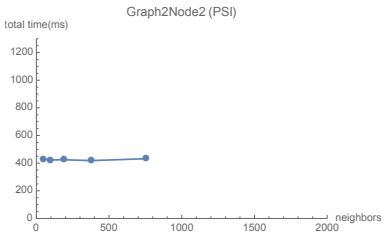
(a) Total run-time: Node 1's neighbours in Graph 1



(b) Total run-time: Node 2's neighbours in Graph 1



(c) Total run-time: Node 1's neighbours in Graph 2



(d) Total run-time: Node 2's neighbours in Graph 2

Figure 6: Total run-time (in milliseconds) of common neighbour on joint graph according to varying sizes of neighbours for Node 1 and Node 2.

We used the Flickr dataset in our experiments and using the SALab tool [3], we generated two graphs based on Flickr. Node and edge similarities are set to 0.5. Graph 1 and Graph 2 has 37377 and 37374 nodes; 1886280 and 1900553 edges respectively. We picked two random nodes, determined their neighbour sets in each graph and ran our experiments using them. In Graph 1, node 1 has 120 neighbours; 48 for node 2 in graph 1; 114 for node 1 in graph 2; and 47 for node 2 in graph 2. Figure 6 shows the total run-time of the common neighbour protocol on the joint graph if we vary the size of neighbours for Node 1 and Node 2 in both graphs.

4.3.2 Utility of the protocol

We illustrate the additional common neighbour information gained by a graph when it collaborates with a second graph. In our first experiment, we created two graphs with the same

[3] GitHub: SALab

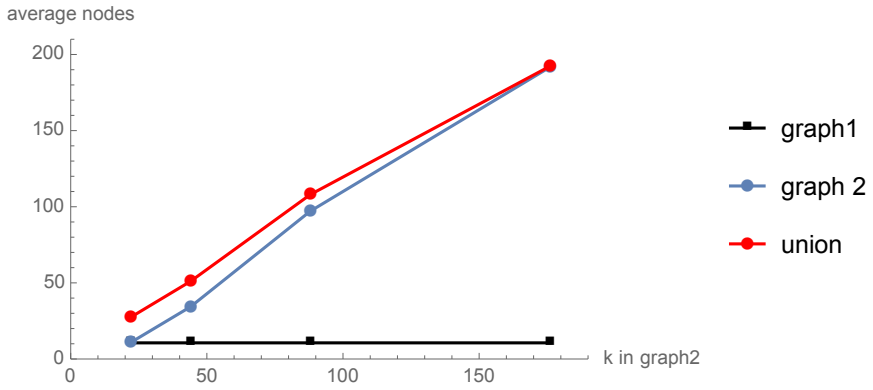


Figure 7: The change in the average number of common neighbours of two pairs according to connectedness of Graph 2. k is the number of edges added at each step in Barabasi-Albert distribution. As k increases in Graph 2, Graph 1 benefits more and more from performing the protocol with Graph 2.

Barabasi-Albert Distribution where 22 edges are added at each step. Both graphs contain 4039 nodes. Average number of common neighbours for each pair is 0.9 in Graph 1. When we consider the merged graph of two networks, average number of common neighbours for each pair increases to 3.3. This shows that distributed link prediction provides significantly more accurate results (compared to local link prediction) and is worth pursuing.

In our second experiment, we created two graphs with the same number of nodes. Both Graph 1 and Graph 2 have 200 nodes. Graph 1 is created with Barabasi-Albert Distribution, where 22 edges are added at each step. Graph 2 is also created in the same setting and, we computed the average number of neighbours of each pair in the union and in Graph 2, with increasing values of k. In this setting, as Graph 2 becomes more connected, it benefits less from the distributed link prediction, as the connectedness of Graph 2 becomes more similar to the union. This is shown in Figure 7.

4.3.3 Security

The privacy and integrity of our proposed solution is largely subsumed by the security of the underlying PSI protocol [42]. This protocol is shown to be secure under the decisional

Graph 1

Vertex	Neighbour set
1	{2,6,7,9,10}
2	{3,8}
3	{2,6,7,8,9,10}
4	{8,9}
5	{6,7,10}
6	{2,3,4,5,8,10}
7	{3,5,10}
8	{2,3,4,6,9}
9	{1,3,4,8}
10	{3,5,7}

Graph 2

Vertex	Neighbour set
1	{4,5,9,10}
2	{5,6,7,9}
3	{8,9,10}
4	{1,9}
5	{1,2,6,7}
6	{2,5,6,7,9,10}
7	{2,5,6,9,10}
8	{3,9}
9	{1,2,3,4,6,7,8}
10	{1,3,6,7}

Figure 8: Sample graphs with 10 nodes. We refer to this sample graph in Section 4.2.1 to explain how common neighbours are computed among two parties and in Section 4.3.3 to quantify the partial information leak.

Diffie-Hellman problem in an appropriate group with semi-honest adversaries. The security proof is in the random oracle model. We make three sequential calls to the protocol. While (universal) composability of the PSI protocol is left by the authors for future work, there is no obvious issue with running the protocol multiple times. For safety, Graph 1 can wait for the first PSI to finish before starting the second one. The PSI protocol leaks (an upper-bound) on the size of each party's graph, and its output is the the cardinality of the intersection. Our protocol, with the three PSI calls, leaks the cardinality of four intermediary values (local2, crossover1, crossover2, and overlap) in computing the common neighbours. We now quantify the impact of this leakage on what Graph 1 can learn about Graph 2 beyond the number of common neighbours.

Leakage of partial information. In our setting, there are three different categories of threats to privacy: identity disclosure, link disclosure, and attribute disclosure. As PSI does not leak the nodes in the set intersection, we do not learn about the identities or the attributes related to the nodes. In PSI [42], the cardinality of intersection leaks information about the possible combination of nodes in the intersection set. Graph 1, who learns the

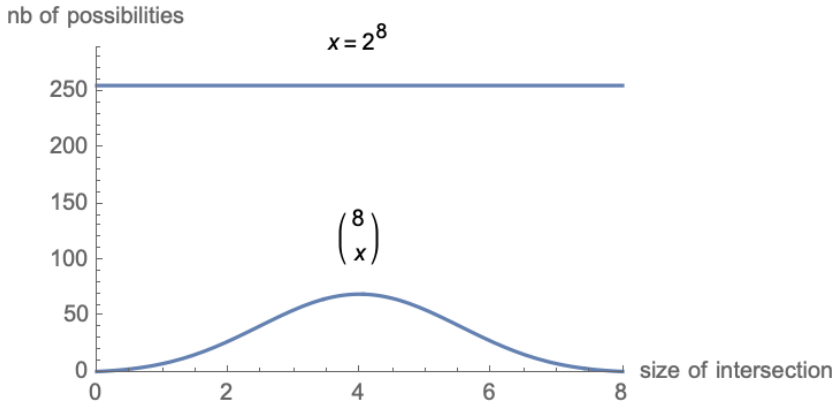


Figure 9: Number of possible combinations of intersection set according to cardinality of intersection.

size of the intersection set, can compute these combinations (which corresponds to link disclosure). The only time Graph 1 learns the identity of a node (which means that the node is in the set that Graph 2 owns) is when Graph 1 has only one node in its set and the cardinality of the intersection it receives as the result of the PSI protocol is 1. Consider the case where Node 1 in Graph 1 has only the node 7 in the set, in Figure 3. When Graph 1 learns that Crossover2 is 1, it can infer that 7 is connected to Node 6 in Graph 2. Therefore, Graph 1 learns the identity of one of the nodes in the neighbour set of Node 6 in Graph 2 and consequently, the link between 7 and Node 6.

We refer to the graphs in Figure 8 in order to illustrate what type of information is leaked during the PSI protocol run between Graph 1 and Graph 2, each having 10 nodes. Graph 1 wants to utilize the graph structure of Graph 2 to decide whether to recommend a link between nodes 1 and 6. At the beginning of the protocol, Graph 2 sends the size of the common neighbours of node 1 and node 6 (which corresponds to local2 and its size is 3) to Graph 1. Hence, Graph 1 learns that there are $\binom{8}{3}$ possibilities for local2 as opposed to 2^8 (we choose from 8 nodes as we assume that 1 and 6 are not neighbours in both of the graphs). When we look at the end cases: (i) if the size of intersection is 0, Graph 1 does not learn any extra information (for node 1, there are 2^8 possibilities with the condition that for each possibility, node 1 and node 6 do not have any nodes in the intersection); and (ii) if

the size of the intersection is 8, Graph 1 learns that nodes 1 and 6 are connected to all of the 8 nodes. Figure 9 illustrates the number of possibilities learned by Graph 1 for each size of the intersection. It also shows that even at the worst case, there is still a lot of information that is not learned by Graph 1.

For a graph generated using the Flickr data set that has 37377 nodes, the average number of neighbours of a node is 50. So, for two nodes with average number of neighbours, there are $\binom{37377}{50}$ possibilities for their intersection, which is a very large number.

4.4 Discussion

4.4.1 Strengthening the privacy

Here, we discuss a solution based on additively homomorphic encryption (*e.g.,* exponential Elgamal or Paillier) that hides all partial information such that Graph 1 learns only the cardinality. At the beginning of the protocol which is shown in Figure 10, Graph 1 determines the neighbour sets of nodes x and y, determines local1 and removes local1 from these sets. It encrypts each element in these sets and the cardinality of local1. Graph 2 performs the same steps for local2. In order to determine crossover1, an encrypted matrix is created: each encrypted element in the neighbour set of node x at Graph 1 is compared against all of the encrypted elements in the neighbour set of node y at Graph 2. The same is done for crossover2 between the neighbour set of y at Graph 1 and the neighbour set of node x at Graph 2. In order to compare two encrypted values, we adapt the protocol proposed in [67]. The comparison function takes two encrypted values and the output is either the encryption of 0 if these encrypted values are different or the encryption of 1 otherwise. The sum over all the elements in the matrix is determined using homomorphic addition both for crossover1 and crossover2. Overlap, which is the intersection of crossover1 and

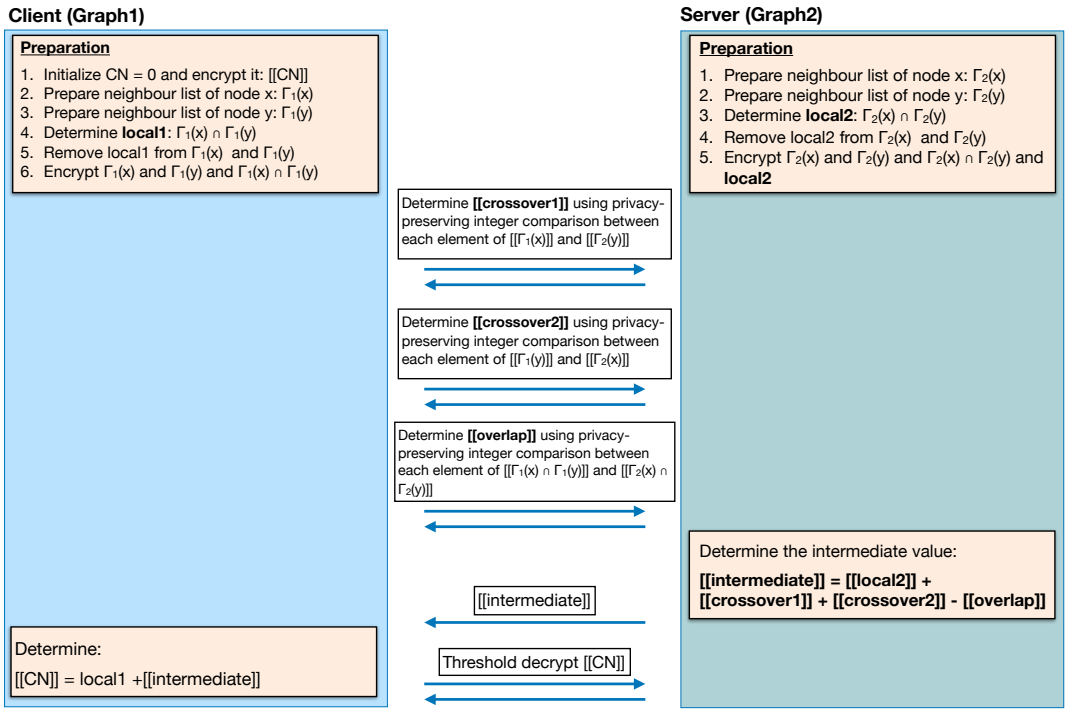


Figure 10: Overview of the solution with stronger privacy. Note that $[[\Gamma_1(x)]]$ means that each element in the neighbour set of node x in Graph 1 is encrypted.

crossover2, is also determined in a similar way using privacy-preserving integer comparison. The final common neighbours cardinality (CN), which is encrypted, is determined as CN= local1 + local2 + crossover1 + crossover2 - overlap. Even though this approach strengthens privacy, it is significantly more costly compared to our proposed solution based on PSI. We think for most applications, the efficiency is a larger concern than the partial leakage from our protocol, but it is possible that entities might prefer complete privacy for very sensitive data.

4.4.2 Complexity

Our protocol's complexity is linear in the sizes of the neighbour set of the nodes. In this paper, we discuss the setting for performing distributed link prediction between two particular nodes in two networks. A network may want to expand this computation for every possible

pair in its graph. The complexity depends on the size (which affects the number of possible pairs) and the density of the network (the sizes of neighbour sets affects PSI run-time). PSI runs in linear time complexity. Our protocol is for a specific pair of nodes. The number of pairs in a graph with n nodes is n^2, so running our protocol (or any protocol based on PSI) one an entire graph will require running a linear time operation on a quadratic number of nodes: thus, cubic time complexity in the worst-case. Reductions in this complexity (*e.g.,* based on heuristics for prioritizing which nodes to look at) is an interesting future work.

4.5 Concluding Remarks

For better accuracy, link prediction can be performed on merged graphs belonging to different parties. This leads to privacy concerns as parties do not want to reveal sensitive data related to their network structures. Therefore, in this work, we proposed a PSI-based, privacy preserving distributed link prediction scheme among two graph databases. In our current proposed scheme, Graph 2 learns among which nodes Graph 1 is computing the common neighbours metric. As a future work, a scheme that allows Graph 2 to hide the identities of these nodes from Graph 1 can be proposed. We might also explore if more recent PSI proposals [73, 26] produce faster results.

We used PSI as a building block for the work explained in this chapter. PSI is much more cost efficient and provides better performance compared to Absentia presented in Chapter 3. The setting for PSI is strictly two-party and the protocol has a lower level of security compared to Absentia. We provide a comparison framework with other approaches to MPC in Chapter 7. In Chapter 5, we present another system that is based on PSI.

Chapter 5

Private set intersection for contact tracing

This chapter is based on the work supervised by Erman Ayday and it is published at 2020 Workshop on Data Privacy Management (DPM) co-located with European Symposium on Research in Computer Security (ESORICS) [45].

In Chapter 4, we looked at using PSI-CA for performing privacy-preserving link prediction between two graph databases. In this chapter, we propose a different application that is based on PSI-CA: privacy-preserving contact tracing.

5.1 Introduction

A pandemic, which typically occurs due to uncontrollable spread of a virus, is a major threat for the mankind. It may have serious consequences including people losing their lives and economical devastation for countries. To decrease the severity of such consequences, it is crucial for countries to track the spread of a virus before it becomes widespread.

Main threat during such a spread is the individuals that had close contact with the carriers of the disease (i.e., people carrying the virus before they are diagnosed with the

disease or before they start showing symptoms). Thus, it is very beneficial to identify and warn individuals that were in close contact with a carrier right after the carrier is diagnosed. If a country can identify who had close contact with the already diagnosed patients, by sending warnings to its citizens and telling them to self-quarantine themselves, the spread of the virus can be controlled. By doing so, individuals that receive such warnings (that they had close contact with one or more diagnosed patients) can take self-measures immediately. This is also economically preferred instead of completely shutting down a country.

However, implementing such an approach is not trivial due to privacy reasons. First of all, due to patient confidentiality, identities of diagnosed patients cannot be shared with other individuals. Similarly, healthy individuals do not want to share sensitive information about themselves (e.g., their whereabouts) with the authorities of the country.

In this chapter, we propose a privacy-preserving technique that allows individuals receive warnings if they have been in close proximity of diagnosed patients in the past few weeks (that is determined based on the incubation period of the virus). We propose keeping the (physical) contact histories of individuals by using communication protocols in their smart phones. These contact histories are then used to determine if an individual was in close contact with a diagnosed patient in the past few weeks, and if so, the individual receives a warning. In order to do this in a privacy-preserving way, the proposed system uses private set intersection (PSI) on the background as the cryptographic building block (between the local contact histories of the individuals and database keeping the identities of the diagnosed patients). Even though PSI consumes more resources, it provides more privacy for the involving parties. By utilizing PSI, we aim to mitigate the privacy vulnerabilities of existing schemes, which will be explained in Section 5.2.

The proposed scheme guarantees that (i) identities or the whereabouts of the diagnosed patients are not revealed to any other individuals, (ii) contact histories of the individuals are not shared with any other parties, (iii) warning received by an individual (saying that

they were in close proximity of a diagnosed patient) is only observed by the corresponding individual and no one else, and (iv) the individual that receives a warning can anonymously share their demographics with the healthcare officials only if they want to. Furthermore, we also propose an extension of the proposed scheme against malicious individuals that may try to tamper their local contact histories in order to learn the diagnosis of some target individuals. We also implement and evaluate the proposed technique to show its efficiency and practicality.

5.2 Related work

The importance of outbreak and disease surveillance (without considering the privacy) has been studied by many previous works [15, 24]. Privacy considerations are addressed in [32], where authors provide a study of existing contact tracing mobile apps for COVID-19. Authors show that none of the existing apps and none of the existing schemes (except for private messaging systems) can protect the privacy of diagnosed patients and other exposed individuals at the same time. However, private messaging systems (in which, a diagnosed patient anonymously sends messages to its previous physical contacts) lack scalability and they heavily rely on proxy servers to obfuscate the identity of the diagnosed patient. Moreover, some countries prefer cell phone tracking-based systems (for diagnosed patients) in order to warn other individuals about the locations of diagnosed patients. However, such systems compromise privacy of individuals to track the spread of a virus [99].

Most of the current proposed systems are Bluetooth-based. Covid Watch [2] is an open source project that relies on Bluetooth for contact tracing and also uses anonymized GPS data to detect high-risk areas. The server keeps contact event numbers from diagnosed users and users compare server's list with their contact list to find out whether they were in contact with a diagnosed person. In [11], a health authority receives information from the diagnosed people and contacts people that were in contact with the diagnosed person. The

health authority keeps record of phone numbers of users. Chan et al. offers functionalities to support the tracking of COVID-19, while also respecting security and privacy requirements by aiming to avoid using trusted third parties [25]. They also discuss the risks of de-anonymization of a user. CONTAIN is another protocol that offers a privacy-preserving solution using Bluetooth [66]. The system does not collect and log privacy-sensitive information. Furthermore, Epione [106] is a PSI-CA based system, where a new semi-honest PSI-CA primitive for asymmetric sets is used. While Epione essentially uses the same protocol as our proposed work, we published our initial idea [44] on arxiv a month before Epione. Epione presents a more theoretical work and it is a slightly faster protocol than ours. In this chapter, we provide full implementation of our proposed protocol. We also believe that the reader might benefit from reading both of the works.

A privacy-preserving Bluetooth protocol for contact tracing is proposed in [3]. Here, each user owns a unique tracing key, from which the user can generate the keys needed for contact tracing. The system relies on Bluetooth for detecting the devices in the proximity. Users receive the information about diagnosed people from a diagnosis server and the matching is done locally on the user's device. In a similar system proposed in [107], users keep their local contact histories by broadcasting their ephemeral, pseudo-random IDs from their smart phones and recording the IDs of other users that are in close proximity. A diagnosed individual voluntarily notifies a server and other users obtain this information from the server. Using this information from the server, the risk of a person for contracting the disease is computed locally on their phone. However, this system is not robust against a malicious user that may try to identify the infected individuals by observing (or modifying) their contact history.

Some apps, including [94, 87] track individuals' location and save it in a local database. However, such an approach compromises location privacy of diagnosed patients since upon diagnosis, location information is gathered by the health authorities. In [94, 87], aggregate

location paths (of diagnosed patients) are sent to the individuals and individuals get notification if they were in close proximity of a diagnosed patient. This results in another privacy vulnerability since individuals can observe the paths of diagnosed patients. Even though only the aggregate location paths are shared by the app, if the number of diagnosed patients is few in a given area, this may result in a significant privacy leak for the whereabouts of diagnosed individuals. As opposed to these approaches, in this work, we propose a scheme that protects the privacy of diagnosed individuals as well as the healthy ones.

Some other works incorporate different building blocks as well. Liu et al. utilize a zero-knowledge protocol to protect privacy of the diagnosed patient and prevent false positive attacks, where either a patient who is not diagnosed with the disease pretend to be diagnosed or a diagnosed patient sends messages to people who are not their close contacts [81]. In [95], a secure multiparty computation-based approach is proposed. Another solution utilizes trusted execution environment (TEE) [1]. Here, users share encrypted location history and also the test status. The system uses private computation to determine the people who were in contact with a diagnosed person. Finally, in [111], authors introduce BeepTrace, which is a blockchain-enabled approach. We summarize different approaches to contact tracing in Table 17.

5.3 Proposed solution

In this section, we first introduce our system and threat models and then, we describe the proposed solution in detail.

5.3.1 System model

The proposed system includes healthy (or not yet diagnosed) individuals with smart phones, diagnosed patients for the disease, and a health authority (e.g., ministry of health or NIH).

Privacy-preserving	Uses ephemeral keys or temporary IDs **Examples:** [11], [3], [107]	**Trust:** The system can be either centralized, where the server does the risk calculation itself and lets the users know about the result, or decentralized, where the risk calculation is done locally at each user's device. For the centralized systems the assumption is that the central authority is trusted. Users can be trusted not to tamper with their local lists or the design of the system prevents the users from changing their lists. **Privacy:** Infected users' privacy and/or healthy users' privacy can be protected based on the system **Performance:** Involves frequent renewal of ephemeral keys or temporary IDs
	PSI-based **Examples:** [106], the work we present in this chapter	**Trust:** No trusted third party is used. If the users are not trusted due to the fact that they may tamper with their local lists, the system can be designed to prevent the users from changing their lists (*e.g.,* by using signatures). **Privacy:** Infected and healthy users' privacy is protected. **Performance:** Involves public key operations
	TEE-based **Examples:** [1]	**Trust:** The hardware is trusted. Third party is used for location collection. **Privacy:** User data is encrypted on the client side and it is transited as encrypted. TEE provides security and privacy during computation. **Performance:** Uses symmetric key operations.
	Blockchain-based **Examples:** [111]	**Trust:** Trusted third parties are utilized in the system. **Privacy:** Relies on encryption and anonymization to protect user privacy. **Performance:** Storage and transactions per second rate affect the performance and the usage of the system at a larger scale.
	Differential privacy-based **Examples:** [87]	**Trust:** User's location is saved in a local database. The authorities that store and aggregate user data continually have to be trusted. **Privacy:** Based on statistical anonymization approaches as opposed to cryptography. **Performance:** Has high performance, as cryptography is not used.
	ZKP-based **Examples:** [81]	**Trust:** The app the users download is trusted. **Privacy:** Users can hide their past location and contact history from the central authority. Both healthy users' and diagnosed users' privacy are protected. **Performance:** Involves modular exponentiations used in digital signature scheme and Σ protocol.
Not privacy-preserving	Outbreak and disease surveillance without considering privacy **Examples:** [15], [24]	**Trust:** The third parties collect and publicly release the data related to disease surveillance. However, whether they are trusted or not is not the main focus of the projects in this category. **Privacy:** Most projects use public data, so the focus is not on protecting the privacy of the users but more on disease surveillance. **Performance:** Focus is more on the accuracy of the results

Table 8: Comparison of different techniques used for contact tracing

Individuals interact with the database of the health authority. In the following, we will describe the proposed scheme assuming a single database for the health authority. For the sake of generality, one can also assume multiple local databases for the health authority (e.g., located in different geographical regions).

The health authority keeps the identities of the diagnosed patients and it does not want this information to be learnt by other parties. Individuals keep their local (physical) contact histories in their smart phones and they do not want this information to be observed by other parties (including the health authority). Also, when an individual receives a warning about a contact with a diagnosed patient, the individual wants to make sure that no other party can observe this warning.

5.3.2 Threat model

We consider a semi-honest attacker model for the parties that are involved in the protocol. That is, each party in the system follows the protocol honestly but they may be curious to learn sensitive information of the other parties. On the other hand, individuals may try to learn the identities of diagnosed patients or the health authority may try to learn the contact histories of the individuals. As we will discuss in detail later, the proposed algorithm protects the parties against these threats. Finally, we assume all communications between parties (between smart phones of two individuals or between an individual and the database of the health authority) are encrypted, and hence robust against eavesdroppers.

In the following, we provide a list of possible attacks against the proposed system.

1. A curious user trying to infer contact information or diagnosis belonging to a target user.

2. A curious user trying to introduce fake contacts in the contact list,

3. The server colluding with a curious user to infer contact information or diagnosis

belonging to another target user.

We discuss how these aforementioned attacks can be mitigated in Section 5.5.3.

5.3.3 Keeping the contact history at local devices

Each individual keeps a vector in their local smart phone for their physical contact history. When an individual A spends some amount of time within the close proximity of an individual B, their phone records the ID of person B. For IDs of the individuals, we propose using the hash of their User ID (UID), which is generated by the application. To measure the proximity between the individuals, we propose using the Bluetooth signals on their devices. Thus, to add individual A as a contact, B needs to spend at least t seconds within r radius of A. This process is also illustrated in Figure 11. As a result of this interaction, the new record in the contact history of A is the ID of B. Each individual may also separately keep the time and the duration of the contact (to have more insight about their risk, as will be discussed later).

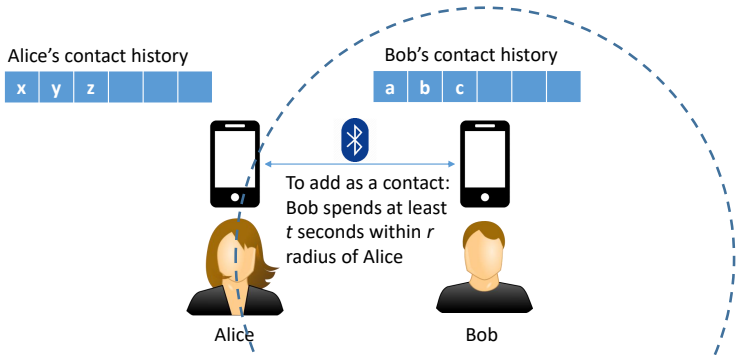


Figure 11: Keeping and updating local contact histories of the individuals

It has been shown that the strengths of Bluetooth signals can be used to approximate the distance between two devices [70, 119]. Alternatively, one can also use (i) just the Bluetooth coverage (e.g., when two devices are in the range of each other for more than t seconds, they can update their local contact histories with each others' IDs), (ii) GPS

information (when two devices are within their Bluetooth coverage, they can exchange GPS information to measure their distance more accurately and update their local contacts if they spend more than t seconds within a close range of each other), or (iii) NFC signal coverage (when the devices are within NFC signal coverage of each other, which is about 3 feet maximum, for more than t seconds, they can update their local contact histories with each others' IDs). Since this part is not the main contribution of this chapter, we do not go into the details of establishing the contact histories.

It is important to make sure that an individual cannot add a contact in their contact list aiming to learn whether a target person has positive diagnosis or not. For instance, knowing the UID of a target person, an attacker may construct its local contact list only from the ID of that target, and hence learn the diagnosis of the target. To prevent such an attack, we consider two options: (i) make sure the local contact histories of individuals are stored in such a way that data cannot be accessed or modified by the individuals (e.g., the local contact history can be encrypted in the device by the key of the health authority or the contact history can be stored in a storage for which the individual does not have read/write permission). Or, (ii) each new contact B of an individual A also includes a digital signature that is signed by a centralized authority (e.g., the telecom operator). To do so, if individuals A and B spend a certain amount of time within close proximity of each other (measured as discussed before), they both send the contact request to the operator, the operator signs and sends back the signed contact record to both parties, and each party keeps the contact records and the corresponding signature together. This way, an attacker cannot fake new contacts in its local contact history. The validity of these signatures are then verified when the local contact history of an individual is compared with the diagnosed patients in the health authority's database (we discuss this in detail in Section 5.5).

Using either of these techniques, the developed algorithm makes sure that the contact history cannot be tampered by the individual. Note that if the storage of the local contact

list would be an overhead, it is also possible to store the local contacts of an individual at a cloud server (encrypted by individual's key) and update the local contacts periodically. Current Bluetooth based solutions, such as [3] do not let the users access their contact histories (that is stored on their mobile devices). Even if the user can access their contact history, tampering with this list can be avoided by using digital signatures and encryption, as explained in this section.

5.3.4 Keeping the IDs of diagnosed patients at a centralized database

When an individual is diagnosed (e.g., by a hospital) with the disease, the User ID (UID) of the positively diagnosed patient is stored in the database of the health authority (e.g., ministry of health), as shown in Figure 12. It is important to note that only the hospital and the health authority know the ID of the diagnosed individual.

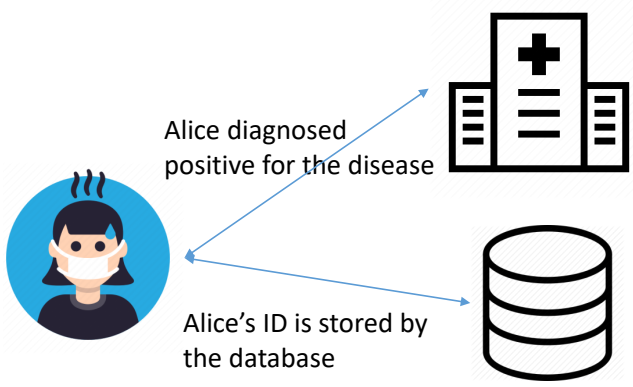


Figure 12: Updating the database of the health authority with the IDs of the diagnosed patients

5.3.5 Private set intersection to identify the individuals at risk

The application on an individual's smart phone sends queries to the health authority's database following a random schedule. This schedule can be determined by the system

to avoid an overload to the database. It is also important not to allow the individual to send queries at any time in order to control the system's bandwidth.

An individual A uses their contact history to query the database of the health authority and the goal is to identify whether there is an intersection between the local contact history of the individual and the IDs of the diagnosed patients in the authority's database (as shown in Figure 13). Size of this intersection reveals the number of diagnosed people with whom A had been in close proximity in the previous a few weeks. As the size of the intersection increases, the risk of individual A being infected also increases. The proposed algorithm provides the result of this intersection to individual A as a warning message. Using the warning, the individual may take early precautions (e.g., have a test or quarantine themselves).

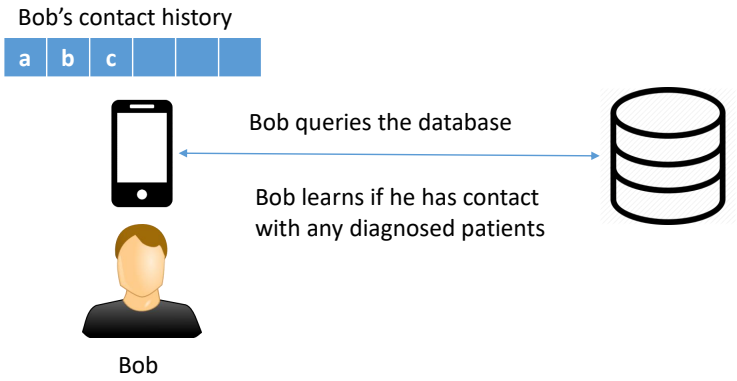


Figure 13: Privacy-preserving interaction between an individual and the database of the health authority

Recall private set intersection cardinality (PSI-CA) from Chapter 2. To compute this intersection in a privacy-preserving way, we use the private set intersection cardinality (PSI-CA) protocol, in which parties that are involved in the protocol obfuscate their inputs (sensitive information) and compute the result of this intersection. Eventually, only individual A learns the result of the intersection and the health authority does not learn any information about the contact history of individual A or the result of the intersection. We also make sure that individual A does not learn anything about the database content of the

health authority (e.g., IDs of diagnosed patients). We provide the details of this protocol in the following.

Figure 14 illustrates the details of the proposed PSI-CA based protocol between an individual (client) and the health authority (server). As input to the protocol, client has its local contact list and server has the list of positively diagnosed patients (steps c.1 and s.1 in the figure).

Client masks its input with the random exponent $R_c^{'}$ and obtains the list of a_i-s and computes $X = g^{R_c}$ (X is similar to an Elgamal public key). Client sends the list of a_i values and X to the server (step c.2 in the figure).

Server permutes its input list and applies $H(.)$ on the list (step s.2 in the figure). Server masks a_i values with its random exponent $R_s^{'}$, shuffles the resulting list, and computes $Y = g^{R_s}$, which is a public-key like value (step s.3 in the figure). Server creates the list of ts_j-s by applying the one-way function $H(.)$ over the multiplication of X^{R_s} and exponentiation of hs_j-s to random value $R_s^{'}$ (step s.4 in the figure). Server sends shuffled and masked a_i values, Y, and ts_j-s to the client.

As the last step, client does the matching between the list of ts_j-s that it received from the server and its own list of tc_i-s (step c.4 in the figure). tc_i-s are obtained by applying the one-way function $H(.)$ over the multiplication of Y^{R_s} and the shuffled a_i-s, which are stripped of the random value $R_c^{'}$ (step c.3 in the figure). At the end of the protocol, client only learns the cardinality of the intersection. At step c.5 in the figure, a notification is generated based on the output of PSI-CA.

5.3.6 Further steps to track the spread

Once an individual receives a warning as a result of the proposed algorithm, they can choose to (i) provide (anonymous) information back to the health authority to help the authority to track the spread and/or (ii) share their local contact history with the health authority to get

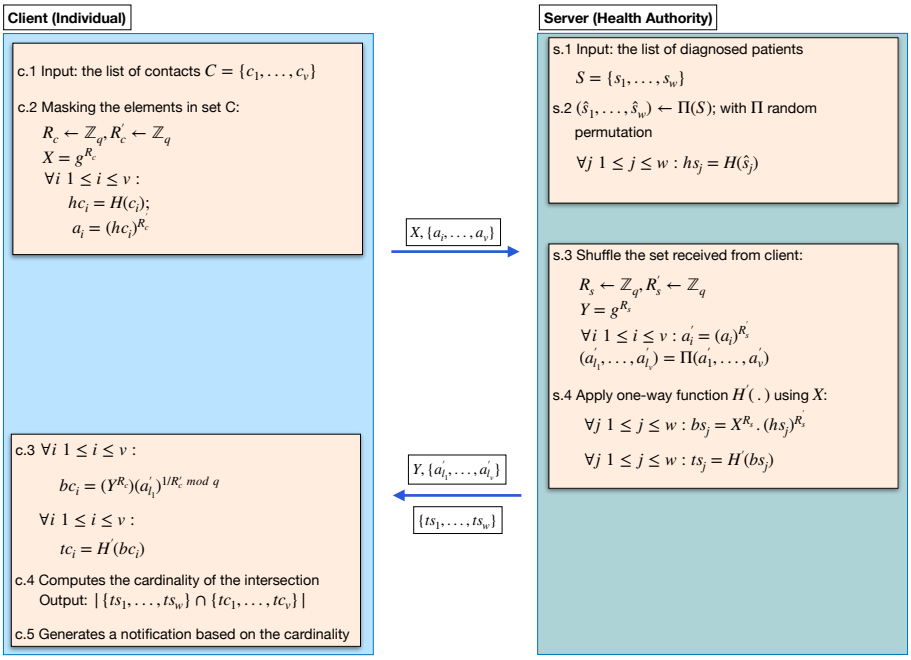


Figure 14: Details of the PSI-CA based protocol between an individual (client) and the health authority (server).

further information about their risk.

In (i), the individual shares their demographics, the size of intersection they obtain as a result of the proposed algorithm, and their location with the health authority. Using such information received from different individuals, the health authority can have a clear idea about how the virus spreads in the population. In (ii), the health authority, using the local contact history of the individual and further details about the contacts of individual (duration and location of each contact), can provide a more detailed risk information to the individual (e.g., if the duration of the contact with a diagnosed patient is long, then the risk of individual also increases). As discussed before, such contact details (i.e., duration or location) are not used in the proposed privacy-preserving algorithm; they can be collected and kept by the individual and may be shared with the health authority to get more insight about the risk.

5.4 Evaluation

We implemented and evaluated the proposed privacy-preserving search algorithm. Our code is open source.[1] We ran our experiments on macOS High Sierra, 2.3 GHz Intel Core i5, 8GB RAM, and 256GB hard disk. We used MD5 as the hash function to hash the UIDs of individuals in the local contact lists and in the health authority's database. We used the implementation of PSI-CA in [42], in which q and p are 160 and 1024 bits, respectively. We ran each experiment for 20 times and reported the average. Note that, if we were to do the same evaluation today, we would compute the confidence interval.

We show the results of the evaluation of PSI-CA based solution in Tables 9 and 10. In Table 9, we set the size of client's local contact list to $1,000$ and vary the size of server's database. In Table 10, we set the size of server's database to $100,000$ and vary the size of client's local contact list. Our results show that the online phase of the protocol can be efficiently completed by the parties even when the input sizes of both parties are significantly large. Note that the server does not need to run the offline part of the algorithm for each client separately. Instead, the server can use the same offline computation during its interaction with every client. Also, a client can conduct its offline steps as it generates its local contact list.

Let's assume that both the server and the client sets are updated in batches. Let B_i represent each active batch (note that the batches expire after the contagious period) in the server's list and C_i represent each active batch in the client's list. Server has batches B_1, B_2, B_3 and client has batches C_1, C_2, C_3. Assume that client and server's list are compared and there are no matches. If there are recently diagnosed patients, the server will update its set as B_1, B_2, B_3, B_4. In this case, C_1, C_2, C_3 can be checked against only B_4. Similarly, if the user has a contact with a new set of user's, the list will be updated as C_1, C_2, C_3, C_4 and only C_4 needs to be compared against B_1, B_2, B_3. While we provided

[1] `https://github.com/ddm2/2020-ContactTracing`

a skecth in this section, the frequency of the updates for client's and server's sets and how the protocol can efficiently do this comparison by reusing the results of the previous comparisons are left as a future work.

Complexity of the proposed algorithm is linear in the size of the two sets. Let the cardinality of server's set be w and client's set be v. Client performs $2(v+1)$ exponentiations with $|q|$-bit (short) exponents modulo $|p|$-bit and v modular multiplications. Server performs $(v+w)$ modular exponentiations with short exponents and w modular multiplications. The resulting communication overhead is $2(v+1)$ $|p|$-bit and w κ-bit values, where κ is the security parameter. It can be deduced from these results that the protocol does not incur a significant overhead for a smart phone. Security and correctness of the proposed algorithm depends on the security and correctness of the original PSI-CA algorithm. We refer to [42] for details.

Table 9: Offline and online run-times for PSI-CA based protocol (in milliseconds) at the client (individual) and server (health authority) with varying size for server's database. Size of client's contact list is set to $1,000$.

Size of server's database	Offline Time (ms)	Online Time (ms)
1,000	Client: 210.6 Server: 388.05	Client: 100.85 Server: 107.5
10,000	Client: 201.2 Server: 2213.5	Client: 978.7 Server: 1003.8
100,000	Client: 202.95 Server: 20054.1	Client: 9766.1 Server: 9925.6
1,000,000	Client: 202.4 Server: 194289.6	Client: 96685.8 Server: 98631.1

5.5 Discussion

In this section, we first discuss an extension of the proposed algorithm, in which we prevent a malicious individual from modifying their local contact history. Then, we discuss about

Table 10: Offline and online run-times for PSI-CA based protocol (in milliseconds) at the client (individual) and server (health authority) with varying size for client's local contact list. Size of server's database is set to $100,000$.

Size of client's contact list	Offline Time (ms)	Online Time (ms)
10	Client: 2.7 Server: 20012.75	Client: 9852.95 Server: 9560.8
100	Client: 22.6 Server: 20218.1	Client: 9854.2 Server: 9968.65
1,000	Client: 202.3 Server: 20448	Client: 9817.5 Server: 9979.75
10,000	Client: 1990.4 Server: 20246.45	Client: 9787.45 Server: 9970.65

Table 11: Offline and online run-times for APSI-based protocol (in milliseconds) at the client (individual) and server (health authority) with varying size for server's database. Size of client's contact list is set to $1,000$.

Size of server's database	Offline Time (ms)	Online Time (ms)
1,000	Client: 1082.25 Server: 508.6	Client: 30.55 Server: 512.55
10,000	Client: 1116.15 Server: 4727.15	Client: 25.3 Server: 483.85
100,000	Client: 1233.15 Server: 46202.1	Client: 49.25 Server: 496
1,000,000	Client: 1019.45 Server: 454721.25	Client: 118 Server: 509.55

potential additional features of the proposed technique.

5.5.1 APSI-based protocol against a malicious individual

As discussed, a malicious individual may tamper with their local contact history to learn the diagnosis of some target individuals. To prevent this, one option is to record each contact along with a corresponding digital signature from a centralized authority, such as the telecom operator (as discussed in Section 5.3.3). Here, we describe how such signatures can

Table 12: Offline and online run-times for APSI-based protocol (in milliseconds) at the client (individual) and server (health authority) with varying size for client's local contact list. Size of server's database is set to $100,000$.

Size of client's contact list	Offline Time (ms)	Online Time (ms)
10	Client: 106.7 Server: 46675.4	Client: 15.75 Server: 6.45
100	Client: 302.4 Server: 46646	Client: 22.75 Server: 61.4
1,000	Client: 1151.05 Server: 45700.35	Client: 46.7 Server: 480.95
10,000	Client: 9507.8 Server: 46946.7	Client: 170.3 Server: 4776.35

be used when computing the intersection between the individual and the health authority's database.

For this, we propose using the authorized private set intersection (APSI) protocol, in which the entries in the local contact history of an individual are digitally signed by a centralized authority and the validity of these signatures are verified by the health authority during the protocol. We discuss the details of this APSI-based protocol in the following.

Recall APSI from Chapter 2. Figure 15 illustrates the details of APSI-based protocol between an individual (client) and the health authority (server) in the semi-honest setting. First, a common input (N, e, g, H, H') is determined for the protocol. $N = pq$ is the RSA modulus, where p and q are safe primes. e is the public exponent. g is a random element in $\mathbb{Z}_N^*$. Also, H and H' are the hash functions, modeled as random oracles. All computations are done in mod N. Both server and the client have the same input sets as in PSI-CA protocol. In addition to these sets, client also has a list of RSA signatures (σ_i)-s, where $\sigma_i = H(c_i)^d \; mod \; N$ (steps c.1 and s.1 in the figure).

As an offline step, server permutes its input list and masks its input. For this, the server first applies the function $H(.)$ over the list of its input elements and exponentiates each element with randomness $2R_s$. Then, hash function H' is applied to the list to obtain ts_j

values (step s.2 in the figure). These values are then sent to the client.

As an online step, client masks σ_i-s by multiplying them with $g^{R_{c:i}}$, where $R_{c:i}$ is the randomness (step c.2 in the figure). Client sends the resulting list that contains a_i values to the server.

At server's online step, server computes $Y = g^{2eR_s}$. Server exponentiates a_i values with $2eR_s$ and obtains the list of $a_i^{'}$-s (step s.3 in the figure). Server sends Y and $a_i^{'}$-s to the client.

At the last step, client obtains the tc_i values by applying the function $H'(.)$ over the product of $a_i^{'}$ and $Y^{-R_{c:i}}$. In order to get the size of the intersection, client finds the matches between the lists of ts_j-s and tc_i-s (step c.3 in the figure). At step c.4, a notification is generated based on the output of APSI.

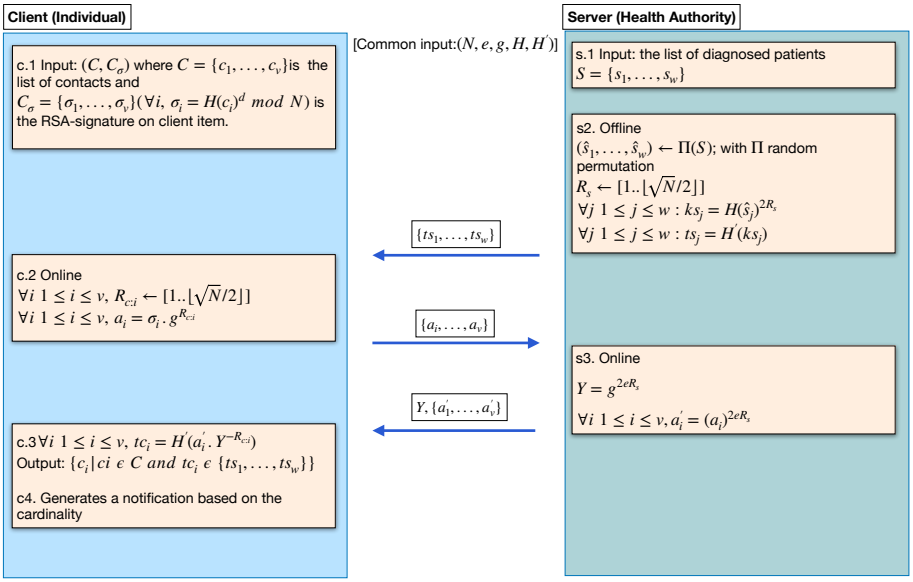


Figure 15: Details of the APSI based protocol between an individual (client) and the health authority (server)

We show the evaluation results of APSI-based solution in Tables 11 and 12. Similar to the setting for the PSI-based solution in Section 5.4, we ran our experiments on macOS

High Sierra, 2.3 GHz Intel Core i5, 8GB RAM, and 256GB hard disk. We used MD5 as the hash function to hash the UIDs of individuals in the local contact lists and in the health authority's database. We used the implementation of APSI in [9], in which q and p are 160 and 1024 bits, respectively. We ran each experiment for 20 times and reported the average. We set the size of client's local contact list to $1,000$ and vary the size of server's database in Table 11. In Table 12, we set the size of server's database to $100,000$ and vary the size of client's local contact list.

We observed that APSI-based solution's online phase can be efficiently completed by the parties, even when the parties have significantly large input sizes. Similar to the PSI-CA based solution, the server does not need to run the offline part of the algorithm for each client separately, as it can use the same offline computation during its interaction with every client. Similarly, a client can perform its offline step using its local contact list.

5.5.2 Additional features of the proposed technique

The proposed technique can also be used to provide real-time whereabouts of diagnosed individuals. Having this information, healthy individuals would know which locations to stay away at any given time. In fact, such an approach has been used by some countries during the recent Coronavirus disease pandemic [83]. However, we believe such a usage of the system may cause social chaos and it may also result in deanonymization of diagnosed patients' identities (even if the information is shared in a differentially-private way [39]). Therefore, we prefer not to include this functionality in the proposed system.

5.5.3 Mitigation against the considered attacks

Here, we discuss the possible mitigation techniques for threats explained in Section 6.4.2.

1. In order to prevent the attack, in which a curious user tries to infer contact information or diagnosis belonging to a target user, a user should not be given read/write

permission to the stored contact list on the device (or the contact list should be kept encrypted in the device). Doing so would prevent an attacker target specific victims by modifying its contact list.

2. To avoid a curious user including fake contacts in the contact list, a digital signature can be included during the contact. The details of this solution are discussed in Section 5.5.1.

3. The attack, in which a server colludes with a curious user to infer contact information or diagnosis belonging to another target user can be addressed as follows: If a curious user can control the contact history, then by colluding with the server, the user can learn the diagnosed people in its contact list. However, if the user cannot control the contact list, the only instance the user can learn whether a person is diagnosed or not is the case where the user has only one person in the contact list. If the user was intentionally in contact with only one person, then the user would know whether that person is positive or not. If the user receives the message that indicates a contact with a diagnosed person, then the user will find out that the person on the contact list is diagnosed with the disease.

5.6 Conclusion

In this chapter, we have proposed a privacy-preserving technique to control the spread of a virus in a population. The proposed technique is based on private set intersection between physical contact histories of individuals (that are recorded using smart phones) and a centralized database (run by a health authority) that keeps the identities of the positively diagnosed patients for the disease. We have shown that individuals can receive warning messages indicating their previous contacts with a positively diagnosed patient as a result of the proposed technique. While doing so, neither of the parties that takes part in the

protocol obtain any sensitive information about each other. We believe that the proposed scheme can efficiently help countries control the spread of a virus in a privacy-preserving way, without violating privacy of their citizens.

Chapter 6

One-time programs for genomic testing

This chapter is based on the work supervised by Kevin R. B. Butler, Mohammad Mannan, Erman Ayday, and Jeremy Clark and co-authored with Lianying Zhao and Joseph I. Choi. It is published at Financial Cryptography and Data Security (FC) 2019 [118].

In Chapter 5, we looked at a contact tracing application that is based on PSI-CA. In this chapter, we present another healthcare-specific application that performs genomic testing using one-time programs.

6.1 Introduction

Consider the well-studied scenario of secure two-party computation: Alice and Bob want to compute a function on their inputs, but they do not want to disclose these inputs to each other (beyond what can be inferred from the output of the computation). This is traditionally handled by an interactive protocol between Alice and Bob, as we have seen in Chapters 4 and 5. In this chapter, we instead study a non-interactive protocol as follows: Alice prepares a device for Bob with the function and her input included; once Bob receives this device from Alice, he supplies his input and learns the outcome of the computation. The device will not reveal the outcome for any additional inputs from Bob (thus, a one-time

program [59]). Alice might be a company selling the device in a retail store, and Bob is the customer; the two never interact directly. By using the device offline, Bob is assured that his input remains private.

A one-time program (OTP) assumes a hypothetical device called one-time memory (OTM) and we show how OTP would be possible if only OTM can be built in the real world. OTM is a very simple device that keeps two keys and only one of these keys is revealed based on user selection, while the other one is destroyed.

OTM is defined theoretically in the original proposal [59], as realizing this device was impractical. Our idea is that Trusted Execution Environments (TEE) seem to offer enough capability to realize OTM. In this chapter, we aim to understand whether TEE can be used to realize an OTM, since TEE guarantees that the program runs as it is defined and it also does not reveal secret values.

Contributions. If we can implement OTM, then we can utilize OTP in a variety of settings:

- We present how genomic testing can be done using one time programs. In this use case, there is a company (Alice) selling devices that performs a genomic test on the customer's (Bob) sequenced genome, without the company learning the customer's genome.
- We give an overview of how OTP can be used in other use cases like database queries, temporary transfer of cryptographic ability, one-time proofs and digital cash.

This chapter aims to find out whether it is possible to use OTM for OTP implementation and if it is possible, whether it is practical to do so. For this purpose, we present two different implementations of an OTP:

- *OTM-based* system where the design of OTP is based on implementing OTM in a TEE. The logic is described as a garbled circuit, where the number of key pairs is

determined by Bob's input size. Hence, the performance of this variant strongly depends on Bob's input size.

- *TXT-only* where we do not use MPC techniques at all. Instead, we rely on the TEE alone to protect the privacy of Alice's input and to run the computation. As Alice's input is directly sealed and unsealed, in this variant the performance is sensitive to Alice's input size.

For each use case, we also discuss which OTP variant is more suitable based on the input sizes of Alice and Bob. Our experiments show that TXT-only has a better performance for the genomic testing use case, as TXT-only does not strongly depend on Bob's input size. Alice can initialize the device in 5.6 seconds and Bob can perform a test in 34 seconds.

6.2 Background and related work

6.2.1 One-time program background

A one-time program can be conceived of as a non-interactive version of a two party computation: $y = f(a, b)$ where a is Alice's private input, b is Bob's, f is a public function (or program), and y is the output. Alice hands to Bob an implementation of $f_a(\cdot)$ which Bob can evaluate on any input of his choosing: $y_b = f_a(b)$. Once he executes on b, he cannot compute $f_a(\cdot)$ again on a different input. For our practical use case, we conceive of OTPs with less generality as originally proposed [59]; essentially we treat them as one-time, non-interactive programs that hide Alice and Bob's private inputs from each other without any strong guarantees on f itself. Note that with a general compiler for f (which we have for both flavours of our system), it is easy but inefficient to keep f private.[1]

[1] Essentially, one would define a very general function we might call Apply that will execute the first input variable on the second: $y = \mathsf{Apply}(f, b) = f(b)$. Since f is now Alice's private input, it is hidden. The implementation of Apply might be a universal circuit where f defines the gates' logic — in this case Apply would leak (an upper-bound on) the circuit size of f but otherwise keep f private.

Recall garbled circuits from Chapter 2. OTP allows two parties to evaluate a garbled circuit in a non-interactive way and one time memory (OTM) is the hardware that performs non-interactive oblivious transfer (OT). Unlike the traditional setting for OT, Bob does not have to interact with Alice to retrieve the key he needs to evaluate the circuit. OTM lets Bob to choose the key only once and the unchosen key is erased, so that Bob does not learn the unchosen key (as a blackbox, it behaves the same as OT). With the help of OTM, one-timeness is enforced and non-interactive OT can be implemented for garbled circuits.

When compared to the traditional setting for OT, OTM has the advantage of being non-interactive and also, Alice does not even have to know who Bob is. On the other hand, OTM has to be physically delivered to Bob and the hardware should be trusted. Thus, one variant is not better than the other, they are just different assumptions (see Chapter 7).

6.2.2 Related work

In the original one-time program paper by Goldwasser *et al.* [59], OTM is left as a theoretical device. In the ensuing years, there have been some design suggestions based on quantum mechanisms [20], physically unclonable functions [75], and FPGA circuits [69]. (**a**) Järvinen *et al.* [69] provide an FPGA-based implementation for GC/OTP, with a GC evaluation of AES, as an example of a complex OTP application. They conclude that although GC/OTP can be realized, their solution should be used only for "truly security-critical applications" due to high deployment and operational costs. They also provide a cryptographic mechanism for protecting against a certain adaptive attack with one-time programs; it is tailored for situations where the function's output size is larger than the length of a special holdoff string stored at each OTM. (**b**) Kitamura *et al.* [76] realize OTP without OTM by proposing a distributed protocol, based on secret sharing, between non-colluding entities to realize the 'select one key; delete the other key' functionality. This

introduces further interaction and entities. Our approach is in the opposite direction: removing all interaction (other than transfer of the device) from the protocol. (**c**) Prior to OTP being proposed, Gunupudi and Tate [62] proposed count-limited private key usage for realizing non-interactive oblivious transfer using a Trusted Platform Module (TPM). Their solution requires changes in the TPM design (due to lack of a TEE). In contrast, we utilize unmodified TPM 1.2. (**d**) In a more generalized setting, ICE [102] and Ariadne [103] consider the state continuity of any stateful program (including N-timeness) in the face of unexpected interruption, and propose mechanisms to ensure both rollback protection and usability (i.e., liveness). We solve the specific problem of one-timeness/N-timeness, focusing more on how to deal with input/output and its implication on performance. We do sacrifice liveness (*i.e.,* we flip the one-timeness flag upon entry and thus the program might run zero times if crashed halfway). We believe their approaches can be applied in conjunction with ours and we leave this as a future work.

6.3 Case study

6.3.1 Genomics background

The genetic instructions that determine development, growth and certain functions are carried on Deoxyribonucleic acid (DNA) [85]. DNA is in the form of double helix, which means that DNA consists of two polymer chains that complement each other. These chains consist of four nucleotides: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). Genetic variations are the reason that approximately 0.5% of an individual's DNA is different from the reference genome.

SNP defines a position in the genome referring to a nucleotide that varies between individuals. Each person has approximately 4 million SNPs. Each SNP contains two alleles, which correspond to nucleotides.

Table 13 lists the SNPs related to BRCA1 and the corresponding risk factors. The magnitude of risk factors ranges from 0 to 10 [100]. A risk factor greater than 3 indicates a significant contribution of that particular allele combination to the overall risk of contracting breast cancer.

SNP Reference Number	Position	Alleles	Risk Factor
rs41293463	43051071	AT	6
		GG	6
		GT	6
rs28897696	43063903	AA	7
		AC	6
rs55770810	43063931	CT	5
		TT	5
rs1799966	43071077	GG	2
		AG	1.1
rs41293455	43082434	CG	5
		CT	5
		TT	2
rs1799950	43094464	GG	2
		AG	1.5
rs4986850	43093454	AA	2
rs2227945	43092113	GG	2
rs16942	43091983	AG	2
		GG	2
rs1800709	43093010	TT	2
rs4986852	43092412	AA	2
rs28897672	43106487	GG	4
		GT	4

Table 13: SNPs on BRCA1 and their corresponding risk factors for breast cancer

Case study We apply our proposed systems on a concrete use case based on genomic testing as a protytype. Single nucleotide polymorphism (SNP) is a common form of mutation in human DNA. Certain sets of SNPs determine the susceptibility of an individual to specific diseases. Analyzing an individual's set of SNPs may reveal what kind of diseases a person may have. More generally, genomic data can uniquely identify a person, as it not only gives information about a person's association with diseases, but also about the individual's relatives [90]. Indeed, advancements in genomics research have given rise to concerns about individual privacy and led to a number of related work in this space. For

instance, Canim *et al.* [22] and Fisch *et al.* [56] utilize tamper-resistant hardware to analyze/store health records. Other works [109, 9] investigate efficient, privacy-preserving analysis of health data.

While a number of different techniques have been proposed for privacy-preserving genomic testing, ours is the first work to address this using one-time programs grounded in secure hardware. Other than providing one-timeness, the proposed scheme also provides (i) *non-interactivity*, in which the user does not need to interact with the vendor during the protocol, and (ii) *pattern-hiding*, which ensures that the patterns used in vendor's test are kept private from the user. On the other hand, homomorphic encryption-based schemes [8] lack non-interactivity and functional encryption-based schemes [89] lack non-interactivity and pattern-hiding. We did not specifically implement these other techniques and compare our solution with them. However, from the performance results that are reported in the original papers, we can argue that our proposed scheme provides comparable (if not better) efficiency compared to these techniques. While in the case of an interactive protocol, where the participants have to communicate back and forth, our proposed system based on OTP is non-interactive. Our system incurs additional cost for hardware, but it can be used completely off-line.

Our aim is to prevent the adversary (the client/Bob), who uses the device for genomic testing, from learning which positions of his genome are checked and how they are checked, specifically for the genomic testing of the breast cancer (BRCA) gene. BRCA1 and BRCA2 are tumor suppressor genes. If certain mutations are observed in these genes, the person will have an increased probability of having breast and/or ovarian cancer [108]. Hence, genomic testing for BRCA1 and BRCA2 mutations is highly indicative of individuals' predisposition to develop breast and/or ovarian cancer.

We also aim to protect the privacy of the vendor (the company/Alice) that provides the genomic testing and prevent the case where the adversary extracts the test, learns how it

works, and consequently, tests other people without having to purchase the test. We aim to protect both the locations that are checked on the genome and the magnitude of the risk factor corresponding to that position. Note that, client's input is secure, as Bob is provided the device and he does not have to interact with Alice to perform the genomic test.

6.3.2 Genomic test

In order to perform our genomic testing, we obtained the SNPs related with BRCA1[2] along with their risk factors from SNPedia [23], an open source wiki site that provides the list of these SNPs. The SNPs that are observed on BRCA1 and their corresponding risk factors for breast cancer are shown in Table 13.

We obtain genotype files of different people from the openSNP website [60]. The genotype files contain the extracted SNPs from a person's genome. At a high level, for each SNP of the patient that is linked to BRCA1, we add the corresponding risk factor to the overall risk.

If a BRCA1-associated SNP is observed in the patient's SNP file, we check the allele combination and add the corresponding risk factor to the total amount. In order to prevent a malicious client from discovering which SNPs are checked, we check every line in the patient's SNP file. If a SNP related to breast cancer is not observed at a certain position, we add zero to the risk factor rather than skipping that SNP to prevent the client from inferring checked SNPs using side channels.

Let i denote the reference number of an SNP and s_i^j be the allele combination of SNP i for individual j. Also, S_i and C_i are two vectors keeping all observed allele combinations of SNP i and the corresponding risk factors, respectively. Then, the equation to calculate the total risk factor for individual j can be shown as $RF_j = \sum_i f(s_i^j)$ where

[2]Similarly, we can also list the SNPs for BRCA2 and determine the contribution of the observed SNPs to the total risk factor.

Algorithm 1 Genomic Algorithm

Input: $RiskFactors, Patient\ SNPs$
Output: RF

```
1: procedure GENOMIC ALGORITHM(RiskFactors, Patient SNPs)
2:     RF = 0
3:     for line in Patient SNPs do
4:         SNP_ID = SNP ID in line
5:         ALLELES = two alleles in line
6:         for line_rf in Risk Factors do
7:             SNP_ID_rf = SNP ID in line_rf
8:             ALLELES_rf = two alleles in line_rf
9:             if SNP_ID = SNP_ID_rf then
10:                if ALLELES = ALLELES_rf then
11:                    RF = RF + risk factor in line_rf
12:                else
13:                    RF = RF + 0
14:                end if
15:            else
16:                RF = RF + 0
17:            end if
18:        end for
19:    end for
20:    return RF
21: end procedure
```

$$f(s_i^j) = \begin{cases} C_i(\ell) & \text{if } s_i^j = S_i(\ell) \text{ for } \ell = 0, 1, \ldots, |S_i| \\ 0 & \text{otherwise} \end{cases}$$

For instance, for the SNP with ID i = rs28897696, $S_i =< AA, AC >$ and $C_i =< 7, 6 >$. If the allele combination of SNP rs28897696 for individual j corresponds to one of the elements in S_i, we add the corresponding value from C_i to the total risk factor.

Genomic algorithm The genomic algorithm is shown in Algorithm 1. Here, RF corresponds to the total risk factor for developing breast cancer. The “risk factors" file contains the associations between the observed SNP and the risk factor, while the “patient SNPs" file contains a patient’s extracted SNPs.

6.3.3 Other use cases

Database Queries. To give an example where the vendor input can be significantly large, we may consider another potential and feasible application of the proposed OTP designs, where OTM-based can outperform TXT-only. It is also in a medical setting where the protocol is between two parties, namely a company that owns a database consisting of patient data and a research center that wants to utilize patient data. The patient data held at the company contains both phenotypical and genotypical properties. The research center wants to perform a test to determine the relationship of a certain mutation (e.g., a SNP) with a given phenotype. There may be three approaches for this scenario:

1. **Private information retrieval [33]:** PIR allows a user to retrieve data from a database without revealing what is retrieved. Moreover, the user also does not learn about the rest of the data in the database (i.e., symmetric PIR [96]). However, it does not let the user compute over the database (such as calculating the relationship of a certain genetic variant with a phenotype among the people in the database).

2. **Database is public, query is private:** The company can keep its database public and the research center can query the database as much as it wants. However, with this approach the privacy of the database is not preserved. Moreover, there is no limit to the queries that the research center does.

   As an alternative to this, database may be kept encrypted and the research center can run its queries on the encrypted database (*e.g.,* homomorphic encryption). The result of the query would then be decrypted by the data owner at the end of the computation [72]. However, this scheme introduces high computational overhead.

3. **Database is not public, query is exposed:** In this approach, the company keeps its database secret and the research center sends the query to the company. This time

the query of the research center is revealed to the company and the privacy of the research center is compromised.

In this use case, the company stores its database into the device (in the form of garbled circuit) and the research center purchases the device to run its query (in TXT) on it. This system enables both parties' privacy. The device does not leak any information about the database and also the company does not learn about the query of the research center, as the research center purchases the device and gives the query as an input to it. In order to determine the relationship of a certain mutation to a phenotype, chi-squared test can be used to determine the p-value, that helps the research center to determine whether a mutation has a significant relation to a phenotype. This use case is left as future work.

Additional genomic tests. Other tests are possible that operate on a sequenced genome. Further, Bob may have multiple inputs to evaluate on a single function. For instance, an individual may input two or more genomes for a paternity test or a disease predisposition test that may also involve other family members. This functionality can be easily added to the proposed scheme by treating multiple sets of test data as single input, although it does not provide privacy between family members (but provides privacy of the set from the vendor).

Temporary transfer of cryptographic ability. OTP lends itself naturally to the situation when one party must delegate to another the ability to encrypt/decrypt or sign/verify messages [59]. In this case, individual OTP boxes must be provisioned and given in advance to the designee, with each box only capable of performing a single crypto-operation. The cost could easily add up, but it might be acceptable for time-sensitive or infrequent messages of high importance (such as military communications). If messages are more frequent, then it may be worthwhile to consider a k-time extension ($k > 1$) to OTP. In either case, the designee is never given access to the raw private key. Care must be taken to restrict the

usable time of each box, which can be realized by sealing an end date in addition to the one-timeness flag.

One-time proofs. As suggested by [59], OTP allows witness-owners to go offline after supplying a proof token to the prover. This proof token can be presented to a verifier only once, after which it is invalidated. We can certainly realize this functionality using our OTP boxes, since proofs produced by our OTP are invalidated by nature of interactive proof systems and may not be reused. Depending on the usage environment, using our OTP box may or may not be cost-effective. While our implemented system may be too costly to serve as subway tickets, the cost may be justified if our box is used as an access-control mechanism to a restricted area.

Digital cash. As a one-time program, this was investigated by [76], which used Shamir's secret sharing in place of OTMs. We borrow their three-party scenario to reason about our own OTP system.

1. The bank supplies OTP boxes with set dollar values.
2. To make a payment, the user provides to the OTP box the shop's hash of a newly generated random number.
    - In TXT, the corresponding keys are selected.
    - After reboot, the selected keys are input into the garbled circuit program, which outputs a signature of the dollar-value concatenated with the shop's hash value.
3. The shop verifies the signature.
4. The shop requests cash from the bank using the signature.

A sealed flag value could enforce the one-timeness, preventing the user from giving valid signatures for more than one shop input. However, our scheme requires further modification to prevent double-spending, as it is possible for two independent shop hash-values to be the same, in which case the user can reuse the associated signature. Furthermore, OTP for digital cash would not be feasible if the held dollar value is less than the cost of the OTP box itself, unless the bank customer's goal is untraceability.

6.4 An OTP implementation

In this section, I provide a high-level overview of the technical solution that is designed and implemented by my co-authors. In order to learn more about the technical details that are not covered in this section, I refer the reader to the full paper [118] and the dissertation of my co-author [117].

6.4.1 Background

Trusted execution aims to protect confidentiality and integrity during a computation. Trusted execution environment (TEE) provides an isolated area for securely executing code. On the integrity side, it lets the user run code and get a quote that the code is exactly the same as what the user is expecting. On the confidentiality side, the code can store private data that can only be unlocked if that exact same code runs again, otherwise it is encrypted with a key that any other program will not have access to.

For the OTP implementation presented in this section Intel Trusted Execution Technology (TXT) is used. Note that, our system can be built for less than $500.[3] TXT uses a tamper-proof hardware that stores data that only the defined program has access to. In order to enforce one-timeness, a flag that is set to 0 is defined and it is stored by TXT. Every

[3] As an example, Intel STK2mv64CC was priced at $499.95 USD on Amazon.com (as of September 2018).

time the program runs, it checks the flag as the first step. The first time it runs (*i.e.,* flag = 0), it sets the flag to 1 and proceeds to running the program. If the program is run again, it checks the value of the flag. If flag = 1, it returns immediately without running further. TXT enforces that the flag can never be reset to 0 (since the code to do this is deliberately not part of program). It also enforces that the check cannot be skipped because the code will always run as coded. If there is a change in the program that is loaded on TXT, the data encoding Alice's input cannot be unsealed and therefore Bob cannot run the computation.

Many papers use SGX as a TEE. We prefer Intel TXT, as it provides exclusiveness. The operating system is a part of TXT, so it does not share hardware with untrusted code. No other code run in parallel once the secure execution starts. This mitigates all software side-channels (see Section 6.6). On the other hand, TEEs that do not provide exclusiveness, like Intel SGX, are more vulnerable to attacks, as trusted and untrusted code share hardware. While Intel TXT provides more security, it is hard to work with it. The instantiation is hard as it does not run on top of an existing OS.

6.4.2 Threat model and requirements

We consider an OTP to be secure for program $y = f(a, b)$ if the following properties are achieved: (1) Alice's input a is confidential from Bob beyond what can be learned from y; (2) Bob's input b is confidential from Alice, and (3) no more than one b can be executed in $f(a, b)$ per device.

Property 1 holds as a is sealed. TXT prevents Bob from looking at a. So he does not learn anything about a that he cannot infer from seeing y. Property 2 is satisfied as the device works off-line. When Bob provides his input to the program, the program runs without interacting with Alice. Property 3 is guaranteed by TEE which enforces one-timeness. In order properties 1 and 3 to be satisfied, there should be no information leakage through software or physical side-channels.

We strive for a reasonable, real-world threat model where we mitigate attacks introduced by our system but do not necessarily resolve attacks that apply broadly to practical security systems. Specifically, we assume:

- Alice is monetarily driven or at least curious to learn Bob's input, while Bob is similarly curious to learn the algorithm of the circuit and/or re-evaluate it on multiple inputs of his choice.
- We assume Alice produces a device that can be reasonably assured to execute as promised (disclosed source, attestation quotes over an integral channel, and no network capabilities).
- We assume that Alice's circuit (including the function and her input) actually constitutes the promised functionality (*e.g.,* is a legitimate genomic test).
- We assume the sound delivery of the device to Bob. We do not consider devices potentially subverted in transit which applies to all electronics [101].
- Both Alice and Bob have to trust the hardware manufacturer (in our case, Intel and the TPM vendor) for their own purposes. Alice trusts the hardware is temper-resistant enough that the circuit can only be evaluated once on a given input from Bob, while Bob trusts that the received circuit is genuine and the output results are trustworthy.
- Bob cannot break cryptographically intractable problems.

6.4.3 System 1: OTM-based

Design and implementation

In System 1, OTP is designed using OTM and garbled circuits [59]. In System 1, TXT is used to only implement OTM functionality. Everything else is done outside of TXT. The garbled circuit is generated in the normal OS environment (*i.e.,* outside of TXT). We use existing tools to generate the wire representation of the circuit (*Frigate* [86]) and to remove the OT step and perform the circuit evaluation (*Battleship* [86]). These are state-of-the-art

for the research literature on GC. As these are designed for garbled circuits, they require modification for OTP. While *Frigate* is used as is, *Battleship* is modified to skip the OT step and instead import the keys from the OTM module in TXT (which just writes them to disk unencrypted).

Figure 16 provides a high-level overview of the system based on OTM. The genomic algorithm (Algorithm 1) and Alice's input (which corresponds to the SNP locations checked for the genomic test) is fed into the *Frigate* compiler to create the wire representation of the algortihm. *Battleship* takes the wire representation to generate the keys and the garbled circuit. Keys are sealed inside a program that runs on TXT and the GC is stored on the device unsealed. The device is than delivered to Bob. He boots in TXT and provides his input (*i.e.,* his sequenced genome). For each bit in Bob's (binary) input, the program makes Bob choose one key and then it makes the other key inaccessible. The keys corresponding to his input are output to the disk. Key selection can occur only once (which is enforced by OTM). After keys are selected in TXT, Bob does not need TXT any more to continue with the evaluation. He uses *Battleship* for circuit evaluation, which is non-interactive and offline. *Battleship* reads the selected key and Alice's input, and prompts the result.

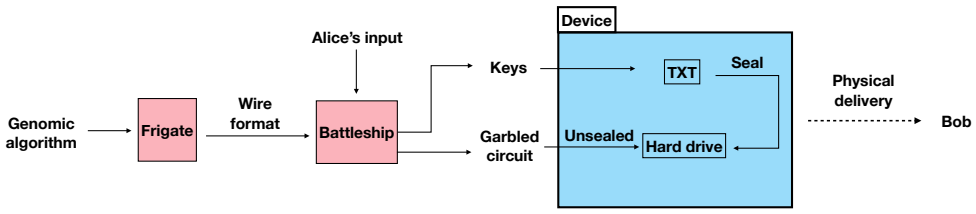


Figure 16: System 1 uses Boolean circuit compiler *Frigate* and its interpreter *Battleship* to generate the garbled circuit [86].

Evaluation

For the genomic case study, Alice (the vendor) has 880 bits and Bob (the client) has 22.4M bits of input. The performance of System 1 is not strongly dependent on vendor's input size.

At vendor's site, it takes around 32 seconds for evaluation. As client input grows in size, more keys must be sealed and this results in an increase in evaluation time for *Battleship* to evaluate the circuit. For 2k bit client input, the evaluation takes around 2 seconds. For 22M bit client input (which corresponds to the genomic use case setting), it takes around 52 minutes.

6.4.4 System 2: TXT-only

Design and implementation

In System 2, the program that performs the genomic test is run only once in TXT. The counter enforces one-timeness. Figure 17 provides a high-level overview of TXT-only system. Alice's input and counter (which is set to 0) are sealed inside the memory. The counter is set to 1 upon entering the OTP, so that the same program cannot be run on another input. The genomic algorithm is kept on the device unsealed. When Bob receives the device, he provides his input (*i.e.,* his sequenced genome) to the device. He loads the TXT program, so that it unseals Alice's input and also takes Bob's input. During evaluation, Alice's input is exposed one record at a time. Therefore, if there is a cold boot attack (a destructive attack where an adversary learns in-memory data only once), the attacker learns only a chunk of Alice's input. As OTP will not run for any other input (the counter is already updated), the attacker cannot recover the rest of Alice's input. Bob's input is secure, as the device cannot leak Bob's input back to Alice without a network connection.

Evaluation

The performance of System 2 does not strongly depend on client's input size up through 224K bits and the execution takes around 9 seconds. After 224K bits, the execution time increases moderately. For 22M bit client input (which corresponds to Bob's input for the genomic test use case) the execution takes 34 seconds. For the setting where the vendor

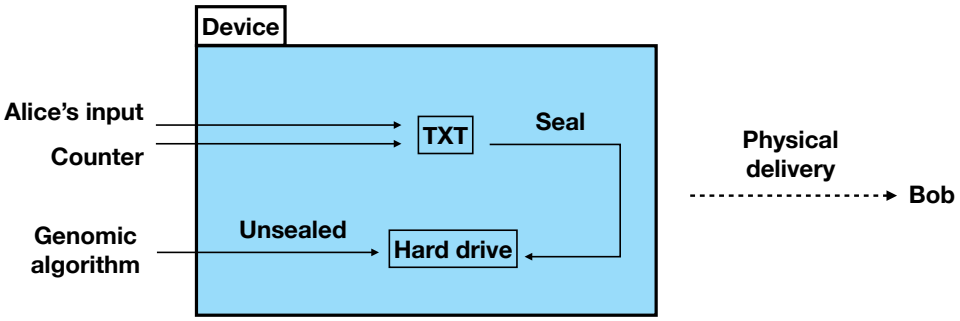


Figure 17: Overview of System 2

input is varied, execution can be performed in 9 seconds for 880 bit vendor input (which corresponds to Alice's input for the genomic test use case) and 921 seconds for 88000 bit vendor input. Unlike System 1, performance of System 2 is dominated by the vendor's input size and less sensitive to client's input. TXT-only is a more suitable choice for the genomic use case, as Bob's input (his sequenced genome) is much larger than Alice's input.

6.5 Choosing an OTP

In the genomic testing use case, Alice's input comprises the 22 SNPs associated with BRCA1(see Table 13). Each SNP entry takes up 40 bits, so Alice's input takes up 880 bits. Bob's input comprises the 701,478 SNPs drawn from his AncestryDNA file, each of which is represented with 32 bits, adding up to a total size of 22,447,296 bits. This genomic test corresponds to the earlier experiment with vendor input size of 880 bits and client input size of 22M bits.

Table 15 puts together the results for both OTP systems. Even at first glance, we see that TXT-only OTP vastly outperforms the OTM-based OTP. Provisioning is two orders of magnitude slower in OTM-based OTP, and trusted selection itself is an order of magnitude slower than the entire execution mode of TXT-only OTP. `gen` (circuit generation) and `evl` (circuit evaluation) further introduce a performance hit to OTM-based OTP. TXT-only is

Small Vendor + Small Client **TXT-only**	Small Vendor + Large Client **TXT-only**
Large Vendor + Small Client **OTM-based**	Large Vendor + Large Client **TXT-only**

Table 14: Depending on the input sizes of vendor and client, one system may be preferred to the other. OTM-based OTP is favorable when large vendor input is paired with small client input; TXT-only OTP otherwise.

OTP Type	Mode	Timing (ms)
TXT-only	Prov.	5640.17
	Execution	33427.50
OTM-based	`gen`	148387.9
	Provisioning	346606.87
	Key Selection	283704.57
	`evl`	3108271

Table 15: Performance for TXT-only and OTM-based OTP implementations of the BRCA1 genomic test, averaged over 10 runs. Vendor input is 880 bits. Client input is 22,447,296 bits.

the superior option for our genomic application.

We already saw in Section 6.4.4 that TXT-only OTP is less sensitive to client input, whereas we saw in Section 6.4.3 that OTM-based OTP is less sensitive to vendor input. We illustrate the four cases in Table 14 in four quadrants.

In this specific use-case of genomic testing, we are in the upper-right quadrant and thus the TXT-only OTP dominates. However, other use cases might occupy the lower-left quadrant, in which case OTM-based will outperform the TXT-only OTP. What should we do if both inputs are of similar size (i.e., equally "small" or "large")? A safe bet is to stick with the TXT-only OTP. Even though GC technology continues to improve, garbled circuits will always be less efficient than running the code natively. For other use cases discussed in Section 6.3.3, we summarize the choice of OTP implementation in Table 16 based on the sizes of vendor and client inputs.

Use case	Vendor input	Client input	OTP implementation
Database queries	large	small	OTM-based
Other genomic tests	small	large	TXT-only
Transfer of cryptographic ability	small	small	TXT-only
One-time proof	small	small	TXT-only
Digital cash	small	small	TXT-only

Table 16: Choosing an OTP implementation for different use cases

6.6 Security analysis

In order to learn the function protected by the TEE, Bob has to buy several devices and run each of them with different inputs. However, this has a high cost and makes this attempt financially prohibitive for Bob.

There can also be attacks on the TEE itself like clonability, replay and side-channel attacks. Cloning the Trusted Platform Module (TPM) is regarded as unfeasible. TPM is a microchip serving as the secure storage and it is one of the hardware components that TXT relies on to function. In a replay attack, OTP is executed several times by replaying a previous state without compromising the TEE. However, it is not possible for the attacker to launch a replay attack as the secrets and the one-timeness indicator are both protected by the TEE (*i.e.,* these values are inaccessible). As for the side-channel attacks, even in the case of a cold-boot attack, the attacker learns a part of the exposed data. As OTP will not run with another input, the attacker cannot learn the rest of the data stored in the device.

There can also be attacks on the cryptography and the protocol itself. The security proofs for OTPs and garbled circuits are covered in the original work [59] (updated after caveat [13]). For more details on these attacks explained in this section, I refer the reader to the full paper [118].

6.7 Concluding remarks

Until now, one-time programs have been theoretical or required highly customized/expensive hardware. We shift away from crypto-intensive approaches to the trusted computing technologies, for a practical and affordable realization of OTPs. With our proposed techniques anyone can build a one-time program today with off-the-shelf devices that will execute quickly at a moderate cost. The cost of our proposed hardware-based solution for a single genomic test can be further diluted by extension to support multiple tests and multiple clients on a single device (which our current construction already does). The general methodology we provide can be adapted to other trusted execution environments to satisfy various application scenarios and optimize the performance/suitability for existing applications.

In this chapter, we presented a genomic testing use case that utilizes OTP. Unlike the previous chapters, this work requires trusted hardware. Like Absentia, it has submit-and-go property. It has the highest performance among other MPC approaches used in this dissertation, as the performance is linear in symmetric key operations. We provide a comparison framework with other approaches to MPC in Chapter 7.

Chapter 7

Conclusion and future work

In this chapter, we return to the main theme that we stated in Chapter 1 and we provide a discussion about comparing the different MPC schemes we used in this dissertation. Finally, we provide possible future directions.

	Submit-and-go	Fully malicious	Publicly verifiable	Cost efficient	No trusted hardware	Identifiable aborts	High performance	3+ parties
Absentia (Chapter 3)	●	●	●	○	●	●	○	●
PSI (Chapters 4 & 5)	○	○	○	●	●	○	◐	○
OTP (Chapter 6)	●	○	○	◐	○	○	●	○

Table 17: Comparison of different approaches to MPC. ●: has the property, ○: does not have the property, ◐: partially satisfies the property

In Table 17, we provide a comparison of different MPC techniques that we used in this dissertation. If a technique has submit-and-go property, it means that the users themselves do not have to participate in the computation. Referring back to the common adversarial assumptions explained in Chapter 1, we determine whether a system provides security

against malicious adversaries. Public verfiability requires that an adversary which arbitrarily deviates from the protocol will be detected by anyone. If a protocol is cost efficient (we assign • for this case), it means that the protocol is running between servers and no special equipment is needed. In OTP, there is no network cost but there is the cost of the additional hardware, hence we assign ◐. A protocol is not cost efficient if the cost is significantly more than the setting where the protocol is run between the servers, and we assign ○ for that system. For instance, in Absentia, the gas cost is way more than the cost of a setting where the protocol is run between servers. We also determine whether a protocol relies on a trusted hardware. If a protocol fulfills the property of identifiable aborts, it means that the participant who aborted can be identified in the case where the computation does not reach finality. If a protocol has high performance, than it is fast and it does not incur high communication cost to participants. If the performance is linear in symmetric key operations, we assign • for the "high performance" column. If it is linear in public key operations, we assign ◐. As for a system that does not have high performance (○ is assigned), there are additional factors like being an interactive protocol with possible delays and containing heavy computations. While some of the protocols are strictly two party, others can be extended to more than two parties.

Absentia, which is based on Mix and Match [68], provides identifiable aborts, payments as an incentive, and security against malicious adversaries, when compared to the other solutions. It can also be generalized to multi-party setting (note that we presented two party setting in Chapter 3). The cost efficiency is lower compared to other techniques, as evaluating a single gate costs thousands of dollars. Absentia does not provide high performance, as Ethereum is slow. For bigger circuits gate by gate evaluation is needed, as the result of the previous gate needs to be computed before continuing with the computation. This increases the cost and evaluating bigger gates has a lower performance. PSI is a low-cost and interactive protocol, with certain amount of information leak about the cardinalities of

the client's and the server's sets. For the use cases presented in Chapter 4 and Chapter 5, the setting is strictly two-party. One time programs we used in the genomic testing use case in Chapter 6 provides submit-and-go functionality like Absentia. However, OTPs do not provide public verifiability and there is an additional cost of the hardware itself, which is not as high as the cost of running the protocol in Absentia.

Having looked at different ways of performing MPC, is one of them preferable to other solutions? We can conclude that not one choice is strictly better from the other, as different applications need different requirements. Every solution uses a different assumption (*e.g.,* trusted hardware for OTP). Cost is also an important factor that affects this choice. Even though Absentia provides properties that other solutions do not fulfill (*i.e.,* operating in the fully malicious setting, public verifiability, identifiable aborts), its cost is currently prohibitive for evaluating bigger circuits. If the setting is known ahead of time, a more specialized MPC fits better. For instance, both use cases in Chapter 4 and Chapter 5 require two-party setting without the need for extending it to a multi-party setting.

We also discuss whether there are different considerations for using an MPC protocol and/or architecture if the protocol is run by individuals as opposed to corporations. The underlying algorithm for an MPC is agnostic to this difference. The determining factor for choosing a suitable MPC is the choices made in the design of the architecture. Some of these choices can be malicious or semi-honest setting, submit-and-go or interactive protocol. Cost is another decision factor. While the individuals are more financially limited, companies can spend more money for the architecture of an MPC protocol. Any MPC protocol that has to keep the parties in the loop for the computation is less preferable for the individuals. A submit-and-go protocol for the personal data is a more logical choice. An interactive protocol can be acceptable for an individual if a specific use case requires one-shot computation. On the other hand, companies can set up servers for recurring computation, which is not a reasonable solution for an individual. When a corporation is involved in the

computation, it generally involves proprietary data (*e.g.,* one-time program that perform genomic testing in Chapter 6). However, there are also use cases where corporation take part in the protocol but the computation is done on personal data. The corporation may be responsible for storing individuals' data and consequently, it is participating in the MPC protocol (*e.g.,* the setting in Chapter 5 where health center keeps the records of individuals).

As we have seen throughout the dissertation, MPC can be used in the settings where personally identifiable information (PII) is involved. Some examples to these settings can be healthcare applications like collaboration of graph database owners to perform link prediction (Chapter 4), contact tracing for tracking the spread of a virus (Chapter 5), or performing a genomic test on the sequenced genome of a user (Chapter 6). Lastly, Absentia is a generic MPC that can be used in any setting where personal and/or proprietary data is involved.

There is another area that we have not discussed in this dissertation: payments. Payments data is both proprietary from the point of view of banks and financial institutions, and it is regarded as personal data by the users participating in the financial system. Exploring the privacy landscape of payments helps us to see the gaps and determine the conflicts between the different stakeholders. Considering these findings a suitable MPC setting can be chosen.

Future work. Every chapter in this dissertation discusses its own future work. Here, we look at possible future work on a higher level. The comparison framework we presented is not comprehensive. A more detailed systematization of knowledge that covers more evaluation categories and other types of MPC protocols can be proposed as a future work. Following a more detailed framework, we can have a more elaborate analysis of the requirements of an MPC protocol depending on the setting it is used.

As for different real-world applications for MPC, even though there are already works in the literature that worked on MPC protocols for payment systems, central bank digital

currencies, which a sub-category, has not been extensively explored yet. A logical first step is to determine the stakeholders participating in the system (*e.g.,* law enforcement and data holders) and determine the conflicts between them. While the law enforcement aims to detect crime, data holders would like to analyze payment data for generating recommendation for users. From the users' point of view, these type of computations should be done in such a way that their privacy is not compromised. We provide some initial exploration on this topic in [7].

Bibliography

[1] Covid-19 self-reporting with privacy, 2020. https://github.com/enigmampc/safetrace.

[2] Covid watch, 2020. https://www.covid-watch.org/article.

[3] Privacy-preserving contact tracing, 2020. https://www.apple.com/covid19/contacttracing.

[4] M. Anagreh, P. Laud, and E. Vainikko. Parallel privacy-preserving shortest path algorithms. *Cryptography*, 5(4):27, 2021.

[5] O. Andreev, B. Glickstein, V. Niu, T. Rinearson, D. Sur, and C. Yun. ZkVM: fast, private, flexible blockchain contracts. Technical report, Online, 2019.

[6] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE, 2014.

[7] R. Auer, R. Böhme, J. Clark, and D. Demirag. Mapping the privacy landscape for central bank digital currencies: Now is the time to shape what future payment flows will reveal about you. *Queue*, 20(4):16–38, 2022.

[8] E. Ayday, J. L. Raisaro, M. Laren, P. Jack, J. Fellay, and J.-P. Hubaux. Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech'13)*, number EPFL-CONF-187118, 2013.

[9] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering gattaca: Efficient and secure testing of fully-sequenced human genomes (full version). *arXiv preprint arXiv:1110.2478*, 2011.

[10] C. Baum, I. Damgård, and C. Orlandi. Publicly auditable secure multi-party computation. In *SCN*, 2014.

[11] J. Bay, J. Kek, A. Tan, C. S. Hau, L. Yongquan, J. Tan, and T. A. Quy. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep*, 18, 2020.

[12] D. Beaver. Commodity-based cryptography. In *ACM STOC*, 1997.

[13] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *ASIACRYPT*, 2012.

[14] D. Bernhard, O. Pereira, and B. Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *ASIACRYPT*, 2012.

[15] S. Bhatia, M. Carrion, E. Cohn, A. Cori, P. Nouvellet, B. Lassmann, L. Madoff, and J. Brownstein. Big brother is watching-using digital disease surveillance tools for near real-time forecasting. *International Journal of Infectious Diseases*, 79:27, 2019.

[16] S. T. Boshrooyeh and A. Küpçü. Inonymous: anonymous invitation-based system. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 219–235. Springer, 2017.

[17] S. T. Boshrooyeh, A. Küpçü, and Ö. Özkasap. Anonyma: Anonymous invitation-only registration in malicious adversarial model. *Cryptology ePrint Archive*, 2019.

[18] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu. Zexe: Enabling decentralized private computation. In *IEEE Symposium on Security and Privacy*, 2020.

[19] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology-ASIACRYPT 2005*, pages 236–252. Springer, 2005.

[20] A. Broadbent, G. Gutoski, and D. Stebila. Quantum one-time programs. In *CRYPTO*, pages 344–360, 2013.

[21] V. Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[22] M. Canim, M. Kantarcioglu, and B. Malin. Secure management of biomedical data with cryptographic hardware. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):166–175, 2012.

[23] M. Cariaso and G. Lennon. SNPedia: a wiki supporting personal genome annotation, interpretation and analysis, 2010. `http://www.SNPedia.com`.

[24] H. A. Carneiro and E. Mylonakis. Google trends: a web-based tool for real-time surveillance of disease outbreaks. *Clinical infectious diseases*, 49(10):1557–1564, 2009.

[25] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine, et al. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544*, 2020.

[26] N. Chandran, D. Gupta, and A. Shah. Circuit-psi with linear complexity via relaxed batch opprf. *Cryptology ePrint Archive*, 2021.

[27] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO*, 1992.

[28] C. Chen, J. Cui, G. Liu, J. Wu, and L. Wang. Survey and open problems in privacy preserving knowledge graph: Merging, query, representation, completion and applications. *arXiv preprint arXiv:2011.10180*, 2020.

[29] H. Chen, Z. Huang, K. Laine, and P. Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1237, 2018.

[30] H. Chen, K. Laine, and P. Rindal. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255, 2017.

[31] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200. IEEE, 2019.

[32] H. Cho, D. Ippolito, and Y. W. Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs, 2020.

[33] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.

[34] J. Clark, D. Demirag, and S. Moosavi. Demystifying stablecoins: Cryptography meets monetary policy. *Queue*, 18(1):39–60, 2020.

[35] J. Clark, D. Demirag, and S. Moosavi. Sok: Demystifying stablecoins. *Available at SSRN 3466371*, 2020.

[36] J. D. Cohen and M. J. Fischer. *A robust and verifiable cryptographically secure election scheme*. Yale University. Department of Computer Science, 1985.

[37] R. Cramer, I. Damgård, and P. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC*, 2000.

[38] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, 1997.

[39] D. Cynthia. Differential privacy. *Automata, languages and programming*, pages 1–12, 2006.

[40] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2013.

[41] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, 2012.

[42] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and private computation of cardinality of set intersection and union. In *International Conference on Cryptology and Network Security*, pages 218–231. Springer, 2012.

[43] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear computational and bandwidth complexity. *IACR Cryptology ePrint Archive*, 2009:491, 2009.

[44] D. Demirag and E. Ayday. Tracking and controlling the spread of a virus in a privacy-preserving way. *arXiv preprint arXiv:2003.13073*, 2020.

[45] D. Demirag and E. Ayday. Tracking the invisible: Privacy-preserving contact tracing to control the spread of a virus. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 240–249. Springer, 2020.

[46] D. Demirag and J. Clark. Absentia: Secure multiparty computation on ethereum. In *International Conference on Financial Cryptography and Data Security*, pages 381–396. Springer, 2021.

[47] D. Demirag and J. Clark. Opening sentences in academic writing: How security researchers defeat the blinking cursor. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 175–181, 2022.

[48] D. Demirag, M. Namazi, E. Ayday, and J. Clark. Privacy-preserving link prediction. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2022.

[49] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.

[50] Y. Dong, J. Tang, S. Wu, J. Tian, N. V. Chawla, J. Rao, and H. Cao. Link prediction and recommendation across heterogeneous social networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 181–190. IEEE, 2012.

[51] Y. Dong, J. Zhang, J. Tang, N. V. Chawla, and B. Wang. Coupledlp: Link prediction in coupled networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 199–208. ACM, 2015.

[52] T. Duong, D. H. Phan, and N. Trieu. Catalic: Delegated psi cardinality with applications to contact tracing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 870–899. Springer, 2020.

[53] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1984.

[54] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[55] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[56] B. A. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov. Iron: Functional encryption using Intel SGX. Technical report, IACR eprint, 2016.

[57] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[58] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, 1999.

[59] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. One-Time Programs. In *CRYPTO*, pages 39–56, 2008.

[60] B. Greshake, P. E. Bayer, H. Rausch, and J. Reda. Opensnp–a crowdsourced web resource for personal genomics. *PLoS One*, 9(3):1–9, 2014.

[61] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais. Sok: Layer-two blockchain protocols. In *Financial Cryptography*, 2020.

[62] V. Gunupudi and S. R. Tate. Generalized non-interactive oblivious transfer using count-limited objects with applications to secure mobile agents. In *Financial Cryptography and Data Security*, FC'08, pages 98–112, 2008.

[63] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols*. Springer, 2010.

[64] X. He, J. Vaidya, B. Shafiq, N. Adam, E. Terzi, and T. Grandison. Efficient privacy-preserving link discovery. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 16–27. Springer, 2009.

[65] J. Heather and D. Lundin. The append-only web bulletin board. In *International Workshop on Formal Aspects in Security and Trust*, pages 242–256. Springer, 2008.

[66] A. Hekmati, G. Ramachandran, and B. Krishnamachari. Contain: privacy-oriented contact tracing protocols for epidemics. *arXiv preprint arXiv:2004.05251*, 2020.

[67] J.-P. Hubaux, J. Fellay, E. Ayday, M. Laren, J. Raisaro, P. Jack, et al. Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech" 13), number EPFL-CONF-187118*, 2013.

[68] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT*, 2000.

[69] K. Järvinen, V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Garbled circuits for leakage-resilience: Hardware implementation and evaluation of one-time programs. In *CHES*, CHES'10, pages 383–397, 2010.

[70] J. Jung, D. Kang, and C. Bae. Distance estimation of smart device using bluetooth. In *The Eighth International Conference on Systems and Networks Communications*, pages 13–18, 2013.

[71] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten. Arbitrum: Scalable, private smart contracts. In *USENIX Security*, 2018.

[72] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin. A cryptographic approach to securely share and query genomic sequences. *IEEE Transactions on information technology in biomedicine*, 12(5):606–617, 2008.

[73] F. Karakoç and A. Küpçü. Linear complexity private set intersection for secure two-party protocols. In *International Conference on Cryptology and Network Security*, pages 409–429. Springer, 2020.

[74] M. Keller, V. Pastro, and D. Rotaru. Overdrive: Making spdz great again. In *EUROCRYPT*, 2018.

[75] M. S. Kirkpatrick, S. Kerr, and E. Bertino. PUF ROKs: A hardware approach to read-once keys. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, AsiaCCS'11, pages 155–164, Hong Kong, China, 2011.

[76] T. Kitamura, K. Shinagawa, T. Nishide, and E. Okamoto. One-time Programs with Cloud Storage and Its Application to Electronic Money. In *APKC*, 2017.

[77] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy*, 2016.

[78] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.

[79] E. A. Leicht, P. Holme, and M. E. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[80] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[81] J. K. Liu, M. H. Au, T. H. Yuen, C. Zuo, J. Wang, A. Sakzad, X. Luo, and L. Li. Privacy-preserving covid-19 contact tracing app: A zero-knowledge proof approach. *IACR Cryptol. ePrint Arch.*, 2020:528, 2020.

[82] W. Liu and L. Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010.

[83] K. Lyons. Governments around the world are increasingly using location data to manage the coronavirus, March 2020. [Online; accessed 26 March 2020].

[84] E. McMurtry, O. Pereira, and V. Teague. When is a test not a proof? In *ESORICS*, 2020.

[85] I. Miko and L. LeJeune. *Essentials of genetics*. 2009.

[86] B. Mood, D. Gupta, H. Carter, K. Butler, and P. Traynor. Frigate: A Validated, Extensible, and Efficient Compiler and Interpreter for Secure Computation. In *Euro-SP*, 2016.

[87] O. S. Moyal. "Hamagen" Application — Fighting the Corona Virus, March 2020. [Online; accessed 26 March 2020].

[88] S. Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[89] M. Naveed, S. Agrawal, M. Prabhakaran, X. Wang, E. Ayday, J.-P. Hubaux, and C. Gunter. Controlled functional encryption. In *CCS*, pages 1280–1291. ACM, 2014.

[90] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. Malin, X. Wang, et al. Privacy and security in the genomic era. In *CCS*, 2014.

[91] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, 1991.

[92] R. Peters. *A secure bulletin board*. PhD thesis, Master's thesis, Technische Universiteit Eindhoven, 2005.

[93] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai. Efficient circuit-based psi with linear communication. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 122–153. Springer, 2019.

[94] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, et al. Apps gone rogue: Maintaining personal privacy in an epidemic. *arXiv preprint arXiv:2003.08567*, 2020.

[95] L. Reichert, S. Brack, and B. Scheuermann. Privacy-preserving contact tracing of covid-19 patients. *IACR Cryptol. ePrint Arch.*, 2020:375, 2020.

[96] F. Saint-Jean. Java Implementation of a Single-Database Computationally Symmetric Private Information Retrieval (cSPIR) Protocol. Technical report, Yale University Department of Computer Science, 2005.

[97] D. C. Sánchez. Raziel: Private and verifiable smart contracts on blockchains. Technical Report 1807.09484, arXiv, 2018.

[98] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptography*, 4, 1991.

[99] K. Servick. Cellphone tracking could help stem the spread of coronavirus. Is privacy the price?, March 2020. [Online; accessed 26 March 2020].

[100] SNPedia. Magnitude. `https://www.snpedia.com/index.php/Magnitude`, 2014.

[101] T. Sottek. NSA reportedly intercepting laptops purchased online to install spy malware, December 2013. Web article. Available at `https://www.theverge.com/2013/12/29/5253226/nsa-cia-fbi-laptop-usb-plant-spy`.

[102] R. Strackx, B. Jacobs, and F. Piessens. Ice: A passive, high-speed, state-continuity scheme. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC'14, pages 106–115, New Orleans, Louisiana, USA, 2014.

[103] R. Strackx and F. Piessens. Ariadne: A minimal approach to state continuity. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 875–892, Austin, TX, 2016.

[104] J. Tang, T. Lou, J. Kleinberg, and S. Wu. Transfer learning to infer social ties across heterogeneous networks. *ACM Transactions on Information Systems (TOIS)*, 34(2):1–43, 2016.

[105] L. T. Thibault, T. Sarry, and A. S. Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 2022.

[106] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293*, 2020.

[107] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, et al. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273*, 2020.

[108] T. Walsh, M. K. Lee, S. Casadei, A. M. Thornton, S. M. Stray, C. Pennil, A. S. Nord, J. B. Mandell, E. M. Swisher, and M.-C. King. Detection of inherited mutations for breast and ovarian cancer using genomic capture and massively parallel sequencing. *Proceedings of the National Academy of Sciences*, 107(28):12629–12633, 2010.

[109] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In *CCS*, pages 492–503. ACM, 2015.

[110] X. Wu, X. Ying, K. Liu, and L. Chen. A survey of privacy-preservation of graphs and social networks. In *Managing and mining graph data*, pages 421–453. Springer, 2010.

[111] H. Xu, L. Zhang, O. Onireti, Y. Fang, W. B. Buchanan, and M. A. Imran. Beeptrace: Blockchain-enabled privacy-preserving contact tracing for covid-19 pandemic and beyond. *arXiv preprint arXiv:2005.10103*, 2020.

[112] Z. Xu, F. Zhou, Y. Li, J. Xu, and Q. Wang. Privacy-preserving subgraph matching protocol for two parties. *International Journal of Foundations of Computer Science*, 30(04):571–588, 2019.

[113] A. C. Yao. Protocols for secure computations. In *IEEE FOCS*, 1982.

[114] K. Yu and W. Chu. Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems*, pages 1657–1664, 2008.

[115] H.-F. Zhang, X.-J. Ma, J. Wang, X. Zhang, D. Pan, and K. Zhong. Privacy-preserving link prediction in multiple private networks. *IEEE Transactions on Computational Social Systems*, 2022.

[116] J. Zhang, P. S. Yu, and Z.-H. Zhou. Meta-path based multi-network collective link prediction. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1286–1295. ACM, 2014.

[117] L. Zhao. *Authentication and Data Protection under Strong Adversarial Model*. PhD thesis, Concordia University, 2018.

[118] L. Zhao, J. I. Choi, D. Demirag, K. R. Butler, M. Mannan, E. Ayday, and J. Clark. One-time programs made practical. In *International Conference on Financial Cryptography and Data Security*, pages 646–666. Springer, 2019.

[119] S. Zhou and J. K. Pollard. Position measurement using bluetooth. *IEEE Transactions on Consumer Electronics*, 52(2):555–558, 2006.

[120] G. Zyskind, O. Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, 2015.